TAPR 20th

ARRL and TAPR DIGITAL COMMUNICATIONS CONFERENCE

CINCINATTI, OHIO

September 21-23, 2001

20th ARRL and TAPR DIGITAL COMMUNICATIONS CONFERENCE

ARRL
225 Main Street
Newington, CT 06111-1494 USA
tel: 860-594-0200 www.arrl.org

Tucson Amateur Packet Radio
8987-309 E Tanque Verde Rd #337
Tucson, AZ 85749-9399 USA
tel: 940-383-0000 www.tapr.org

Printed in USA

ISBN: 0-87259-852-7

ARRL Order Number: 8527

First Edition

Welcome!

The 20th ARRL and TAPR Digital Communication Conference marks a two-decade milestone in the advancement of state-of-the-art Amateur Radio technology.

These proceedings dramatically illustrate how far we've come in 20 years. APRS has "conquered" the amateur packet world, and the APRS-compatible weather station discussed in these pages demonstrates the practical, public-service side of this innovative mode. As commercial television moves into the digital era, amateur TV may soon follow based on the article you'll read here. You'll also read how hams continue to explore ideas relating to the development of HF digital voice. Other fascinating topics include TNC firmware, TCP/IP and packet Internet e-mail.

I hope these proceedings and this conference will inspire you to begin some digital explorations of your own—and that we'll have the pleasure of seeing one of your articles in the 2002 edition.

David Sumner, K1ZZ
ARRL Executive Vice President

September 2001

Table of Contents

μWeather™ – An APRS-compatible weather station

David R. Andersen, K0RX
63 Pleasant Hill Road
Mt. Vernon, IA 52314

k0rx@arrl.net
http://www.qsl.net/k0rx/uWx

Abstract

In this paper, I describe an APRS-compatible, single-board weather station project that I have developed. The weather station is based on the Microchip 16F877 microcontroller. The basic weather station monitors temperature, relative humidity, and barometric pressure. These data are periodically formed into APRS weather packets sent using on-board AFSK generation. Thus, no TNC is required to transmit the weather packets over the air. In addition, the data are sent in industry-standard "datalogger" format out the serial port for use with any of several weather station monitoring programs. Future plans call for implementing the ability to upgrade the weather station by attaching an optional rain gauge and/or anemometer. Any rain gauge that emits a TTL pulse for each 0.01 inch of rain received as well as the Dallas Semiconductor 1-wire anemometer will work with μWeather™.

Introduction

In the last few years, the Automatic Packet Reporting System (APRS) has become one of the most popular amateur packet radio applications in existence. APRS links together a wide variety of different telemetry-producing devices such as weather stations, GPS trackers, and balloon and model rocket telemetry systems into a tremendously interesting and dynamic network of amateur radio enthusiasts. The public service applications of this network are widespread – helping to track everything from storm spotters to parades and automobile rallys, and providing a comprehensive set of weather data that is uplinked to a web page for retrieval by all interested parties are just a sample of the services that APRS provides. Recently, the National Weather Service (NWS) has begun to make use of this data, incorporating it into a NEMOnet page on the web that provides a comprehensive initial data set for weather prediction models used by the NWS to issue the forecasts that we all depend on. These public-service applications are precisely what makes APRS interesting and fun for the legion of ham radio operators that participate in this aspect of our diverse hobby.

We have all had the experience of watching the weather forecast on the television news report, and waking up the next morning to a completely different weather picture than what was predicted the night before. Such a situation can be frustrating at best, and dangerous at worst. In fact, I find the weather-reporting capabilities of APRS aid me in my own predictive efforts – and believe that I can do at least as good of a job as the pseudo-professionals on TV. However, there are difficulties with the weather data supplied by the APRS network. These difficulties arise because of two basic characteristics of commercial APRS-compatible weather stations: they are expensive and complex. Currently, a simple weather station reporting temperature, barometric pressure, and humidity (in my opinion the three most useful sets of predictive weather data) requires a basic station plus possibly additional sensors, a computer or other conversion device, and a TNC. All told, the costs of a basic weather station can easily climb into the multi-hundred dollar range, and that doesn't include the radio that actually sends the packets. As a result of the cost and complexity of these stations, it is simply not possible to deploy as many simple, reliable weather stations as is required for accurate forecasting.

Will Beals, N0XGA, and Russ Chadwick, KB0TVJ, have developed an excellent experimenters weather station kit for APRS use, the T238. This weather station is available through TAPR, and the necessary sensors, originally developed by Dallas Semiconductor, Inc. can be obtained quite easily as well. The T238 project is a big step in the right direction, with a significantly lower cost than commercially-available weather stations of comparable accuracy. However, this kit does not support the on-board generation of AFSK tones, and as a result an external TNC is required for APRS applications. The fact that all sensors are off-board, coupled with the extra expense and complexity required with the use of an external TNC makes this weather station less suitable for remote applications.

My goal in designing μWeather™ was to develop a weather station that would fulfill the need that I identified above. The station had to be simple, low-cost, and capable of accurately reporting weather data in a wide variety of deployment scenarios. I had been following the work of John Hansen, W2FS (PIC-based KISS TNC, http://john.hansen.net) and Byon Garrabrant, N6BG (TinyTrak, http://www.byonics.com/tinytrak), and thought that techniques similar to ones they had used for their APRS/Packet Radio projects could be used in a low-cost microcontroller-based weather station. The weather station described in this paper is the result of my development efforts, as well as able assistance from beta-tester Tony Arnerich, KD7TA.

Weather Station Concept

Initially, the concept for my weather station was quite simple – I wanted to be able to monitor temperature, barometric pressure, and humidity. To do this, the weather station would need to monitor the sensors and then generate AX.25 packets and the associated audio-frequency shift keying tones to send them. However, as in some of J. R. R. Tolkien's stories, this project grew in the development process. A serial port would be needed for the initial configuration of the weather station, including the station's callsign, a list of digipeaters, APRS parameters such as the station's location (latitude and longitude) and various transmission parameters such as the TXD setting. At some point during the conceptualization phase of the project, it occurred to me that it would be pretty straightforward to add serial datalogger output. After all, I already had the data,

and it was just a matter of formatting it properly. In addition, after conversations with a couple of friends of mine about the feature list for this station, I decided to add an LCD display, and rain gauge and anemometer capability as well. Thus, the basic station quickly transformed itself into something resembling a full-featured weather station. However, I've worked hard to make sure that the original design goal of creating a single-board weather station that is capable of monitoring very basic weather data and transmitting that over the APRS network using the AX.25 packet radio protocol has survived. The LCD display, datalogger monitoring, rain gauge, and anemometer are all options that do not have to be incorporated.

With the dust settling on the design phase of this project, μWeather™ has the following feature list:

- Single-board weather station microcontroller-based weather station
- On-board or remotable temperature/humidity/bar. pressure sensors
- Real-time clock for time-stamping APRS weather packets
- On-board generation of AFSK for APRS weather packets (no TNC required) with standard 5-pin DIN receptacle for interfacing with radio
- Weather data is also output in RS232 industry-standard datalogger format for compatibility with 3rd-party weather display software
- 20x4 character LCD for continuous display of current weather conditions
- Easy initial configuration performed through the μWeather™'s RS232 port with any standard terminal or terminal emulator software
- Future enhancements - inputs for optional rain gauge and anemometer

Design Description

The first step in implementing this project was to determine what microcontroller I was going to use. When I began the project, I had very little experience working with any microcontroller, so the field was pretty open. After examining the options, I settled on the Microchip 16F877 for three basic reasons: 1) this microcontroller has on-chip analog to digital conversion capability which would help reduce the chip count and therefore the cost, 2) it is flash programmable which would reduce the development time, and 3) it is based on the concept of reduced instruction set computing (RISC) which

simplified the coding process. I wanted to write the firmware in assembly language so this third item was a key consideration in my mind.

In order to achieve the specified design functionality, I needed to incorporate three sensors on the weather station board. These were temperature, barometric pressure, and relative humidity sensors. In addition, since weather data without a time-stamp is of little value, a real-time clock would be required. For temperature measurement, I chose to use the DS1621 digital thermostat, an 8-pin I^2C device. As a result, I opted for the 8-pin DS1307 real-time clock in order to use the same I^2C bus that I had already incorporated in the design. For the measurement of barometric pressure, I chose the MPX4115A sensor by Motorola, and for relative humidity measurement, I used the HIH-3610. Both of these last two sensors put out a voltage that is proportional to the parameter they are measuring, and so are read using the on-chip A/D capability of the 16F877.

The most difficult of these three sensors to incorporate in the design was the barometric pressure module. The 16F877's on-board A/D has 10-bit resolution, which means that for A/D voltage references of 0 and 5V respectively, the MPX4115A could be read with a resolution of about 1.1 mBar. This was an order of magnitude poorer resolution than the APRS specification permits. However, the 16F877 has a facility for using voltage references other than the power supply voltages in conjunction with it's A/D module. Thus, through the use of judicious hardware design, plus digital filtering techniques in the firmware, it was possible to enhance the barometric pressure resolution to approximately 0.1 mBar: exactly what the APRS specification permits.

On-board constant-phase AFSK tones are generated using a 4-bit resistive ladder D/A converter. This approach uses a standard technique that is well-documented in Microchip application note AN655, and waveforms generated in this manner were used successfully in the TinyTrak APRS tracker by N6BG. I generate my AFSK tones using delay tables, and the bit period is fixed using the TMR0 interrupt facility of the 16F877.

Configuration data is input to and serial datalogger data is output from the 16F877 through a bit-bang RS232 serial port in 2400 baud 8-N-1 format. I chose to use my bit-bang routines, rather than the built-in UART on the 16F877 primarily because my early development work had been done on a 16F84 microcontroller that did not have an on-

board UART. However, the the TX and RX pins for the built-in serial port have not been used elsewhere in the weather station design in order to permit the use of a simple boot-loader for firmware upgrades in the future. The 2400 baud rate was chosen in order to be compatible with the industry-standard datalogger output format. In order to maintain the design philosophy and keep the chip count down, I have not used a level-converter, and the voltage levels are TTL. This may give rise to compatibility issues with some computer serial ports, however this has not been a problem to date.

I selected an industry-standard 4x20 HD44780-based LCD text module for the display. This permits me to output current weather data in a user-readable format with a relatively low cost. Such display modules are available surplus for very little money. In addition, the LCD interface is designed such that removing the LCD for remote operation will not affect the functionality of the weather station. One of my future plans for enhancement is to substitute an LCD graphics display for the text module so that I can plot historical weather data as well. Such a modification would greatly enhance the utility of the weather station for stand-alone weather prediction efforts.

Currently, approximately 4K of the 8K available 16F877 program space has been used by the μWeather™ firmware. This leaves plenty of room for enhancements such as the graphics display and the rain gauge/anemometer. Also, μWeather™'s current draw is less than 15 mA with the LCD attached, indicating that battery/solar-powered operation is feasible.

Summary

In summary, this paper describes the development of a low-cost, APRS-compatible, single board weather station suitable for remote deployment. The station is based on a Microchip 16F877 flash microcontroller. The station monitors and periodically formats temperature, barometric pressure, and humidity data into APRS weather packets and transmits them using AFSK tones generated on-board. The station also outputs the weather data in industry-standard serial datalogger format. Up-to-date information concerning μWeather™'s development status may be obtained at the project's web page: http://www.qsl.net/k0rx/uWx.

Acknowledgements

1) μWeather™ is a trademark of David R. Andersen, K0RX.
2) APRS is a trademark of Bob Bruninga, WB4APR.
3) The hard work of this project's beta tester, Tony Arnerich, KD7TA is greatfully acknowledged. Without his efforts, the project would be much less than it is.

Both Way Radio Internet Email

John Blowsky, KB2SCS kb2scs@arrl.net

Abstract:
BWRIE is a software system that allows the user to send and receive Internet Emails via Amateur Packet Radio. BWRIE consists of two plain vanilla AX25 packet radio stations. One station runs my "Send" software the other runs my "Receive" software. The Receive station also has an Internet connection. A Full Time or Dial Up connection, either will work.

Key Words:
BWRIE, Amateur Packet Radio, Emergency, Desaster.

Why Was BWRIE Created?
BWRIE was created to fill the perceived need of sending Internet Emails during an emergency. Yes the Internet is down inside the desaster area. But outside of this area the Internet is doing just fine. All that is needed is a Receive station be set up outside where the Internet is working. Naturaly also a Send station would be needed inside the desaster area. If Send and Receive can reach eachother via RF then this system will work very well.

Receive Station Hardware:
The RECEIVE packet station consists of the following:

1) Computer:
   Capable of running Windows 95 or better.
   BWRIE was tested using Second Edition Windows 98.
2) TNC :
   BWRIE was tested using a Kantronics KPC3+
   Note the RECEIVE packet station has to have a TNC capable of doing full HARDWARE control. In other words a full 9 wire serial cable has to be used.
3) Internet Connection:
   Full time or dial up is ok. Note BWRIE was tested using a full time Internet connection. BWRIE does not have proxy support. In the future if there is demand then proxie support will be Added.
4) Transceiver:
   Any FM transceiver will work. BWRIE was tested on 2M using a Yaesu FT-5100.

Send Station Hardware:
1) Computer:
   Capable of running Windows 95 or better.
   BWRIE was tested using Second Edition Windows 98.
2) TNC:
   BWRIE was tested using a Kenwood D700.
   Note the SEND packet station's TNC can be any TNC. A two-wire type of serial connection is fine. Which means you can use a Kenwood THD7 for the SEND station.
3) Transceiver:
   BWRIE was tested on 2M. As stated above a Kenwood D700 was used.

How Receive And Send Work Together:

Please refer to Figures 1 through 4. Which are found at the end of this paper.
SENDING EMAILS:
Once the user has the RECEIVE and SEND Stations up and running and a Packet Radio connection is established between SEND and RECEIVE. The next step is for the operator of the SEND Station to fill in the "Email" fields and then click on the Send button. When the Send button has been clicked the message is first saved to the SEND Stations hard drive. This is done incase there is a problem with the packet connection. This way if the message is lost during the packet transmit phase then at least the message is saved and can be sent manualy when the packet connection is restored. Cutting and pasting the file into the txtmessage field can do this.
The files are stored in the C:\Program Files\BWRIE directory.
The first Email that Send sends is named 1.txt. The second file is named 2.txt and so on.
Note each time you start up Send send starts with file name 1.txt which of course means that the old 1.txt gets written over. This was done so that your hard drive would not run out of room and old not needed files would be automatical removed.
After SEND has saved the message locally then SEND sends the message to RECEIVE via Packet Radio.
If all goes well with the packet phase of the process the first thing RECEIVE does is save the message to its hard drive. For the same reason as above. You may have lost your Internet connection.
After RECEIVE has saved the message it then formats it into an Internet Email message and then it uses **S**imple **M**ail **T**ransfer **P**rotocol to send it out to the Internet.
At this point RECEIVE then sends a Packet Radio message to SEND stating its success or failure with sending the Email message. If success then the status field in SEND contains
"Mail Sent Successfully"
If RECEIVE is not successful then the status field in SEND contains
"Mail Sent". Note the missing word SUCESSFULLY.
Note: Once Receive is setup and is running it can be run automaticaly with no operator intervention. Running Receive in this mode is recommended for uses such as Field Day. Where if the Internet connection is lost it is easy for the send operator to go home and restore his or her Internet connection. It is recomended for an operator to be at the "controls" of the Receive station during emergency mode of use. This way if the Internet connection is lost the receive operator can restore the Internet connection and then send the Email manualy. Just like with send above.

Receiving Emails:

Once the user has the RECEIVE and SEND Stations up and running and a Packet Radio connection is established between SEND and RECEIVE. The next step is for the operator of the SEND Station to click on the "Getmail" button.
Clicking on this button causes Send to send a request to Receive asking for a list of the Emails that Receive has waiting to be sent to Send.
When Receive receives this request it sends the list of waiting Emails to Send.
When Send receives this list, Send displays this list on its screen.
The Ham at the controls of SEND can read this list and then by double clicking on a list item would then cause SEND to send a request for this particular Email to Receive.
When Receive receives this request for this particular Email, Receive would then send this Email to SEND.
When Send receives this Email, Send will display the Email on its screen.

How Do I Get BWRIE?

BWRIE can be downloaded from the following URL.
http://www.qsl.net/kb2scs

Figure 1

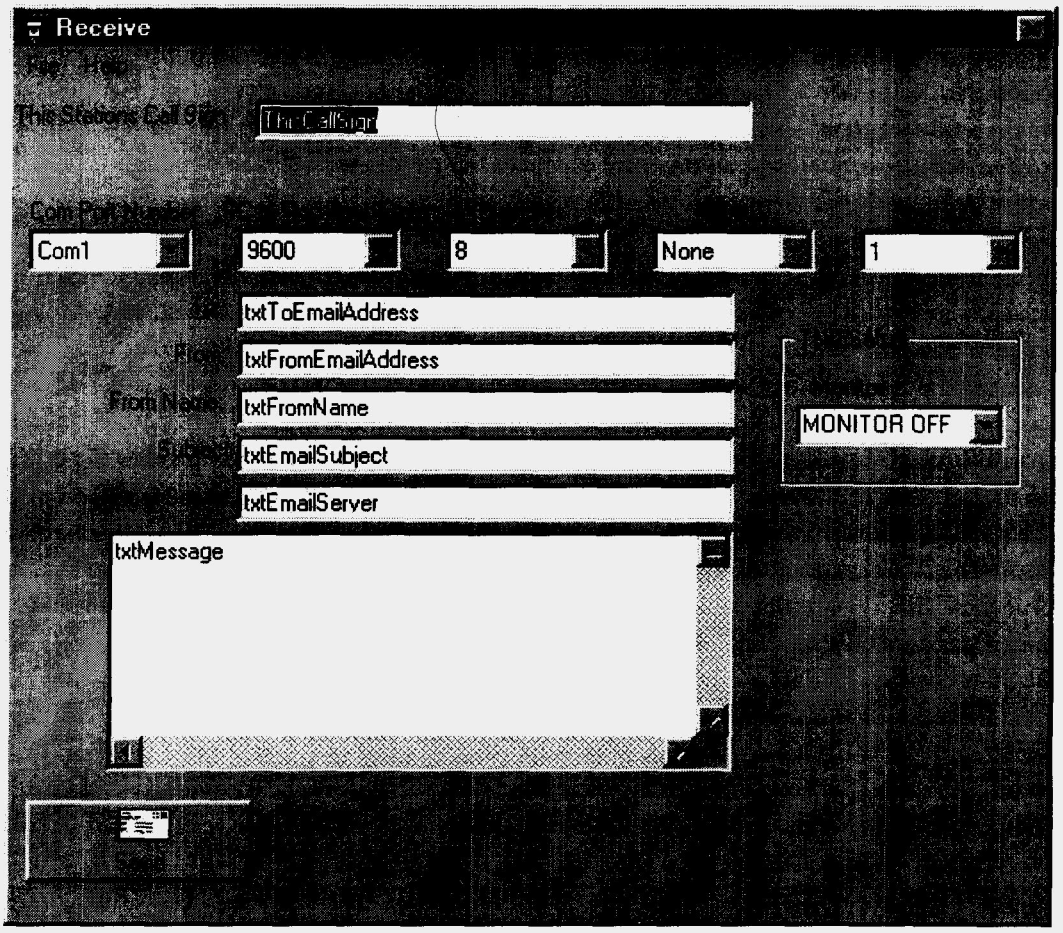


Figure 2

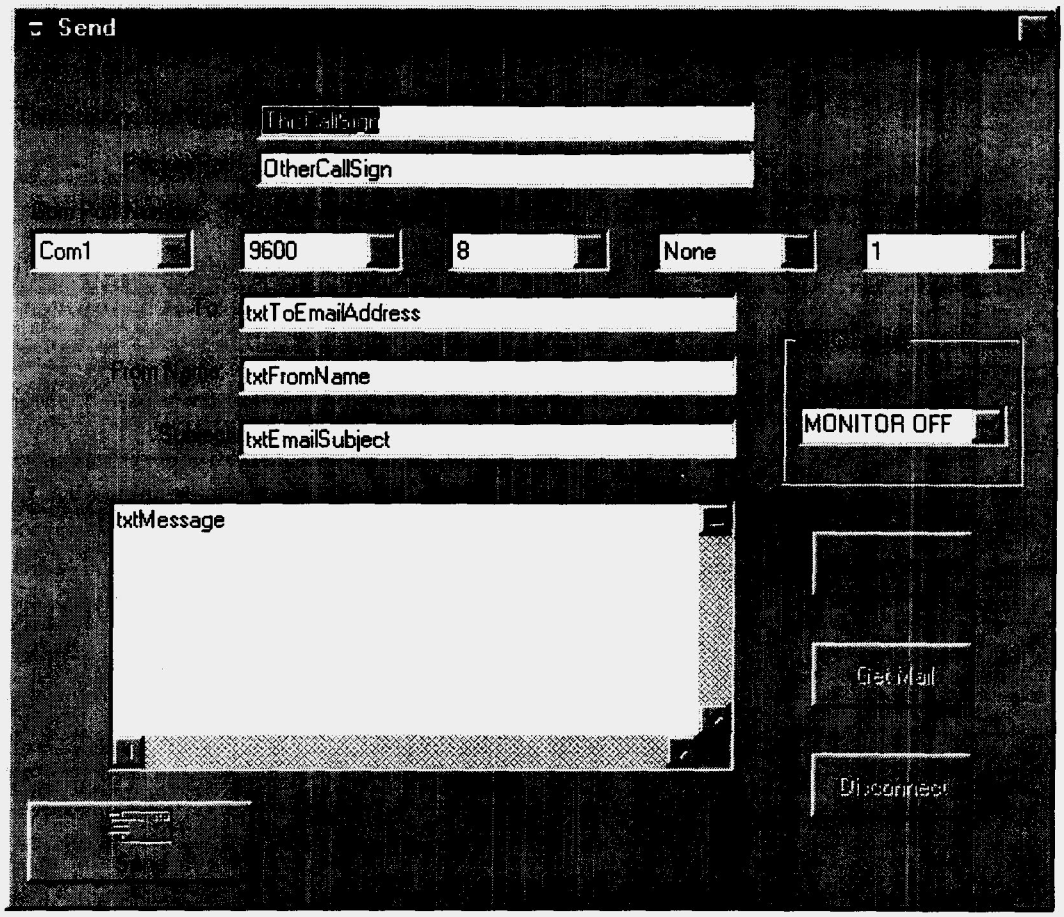

Send
OtherCallSign
Com1
9600
8
None
1
txtToEmailAddress
txtFromName
txtEmailSubject
MONITOR OFF
txtMessage
Get Mail
Disconnect

Figure 3

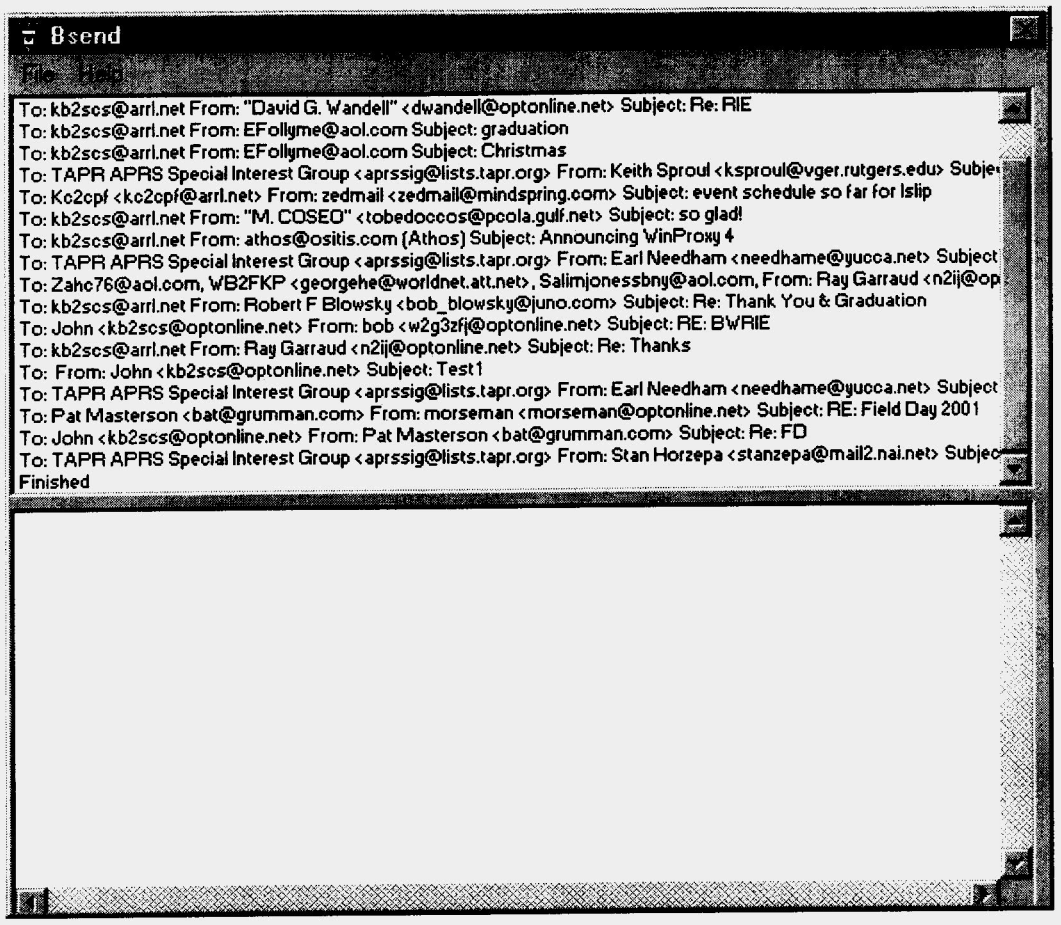
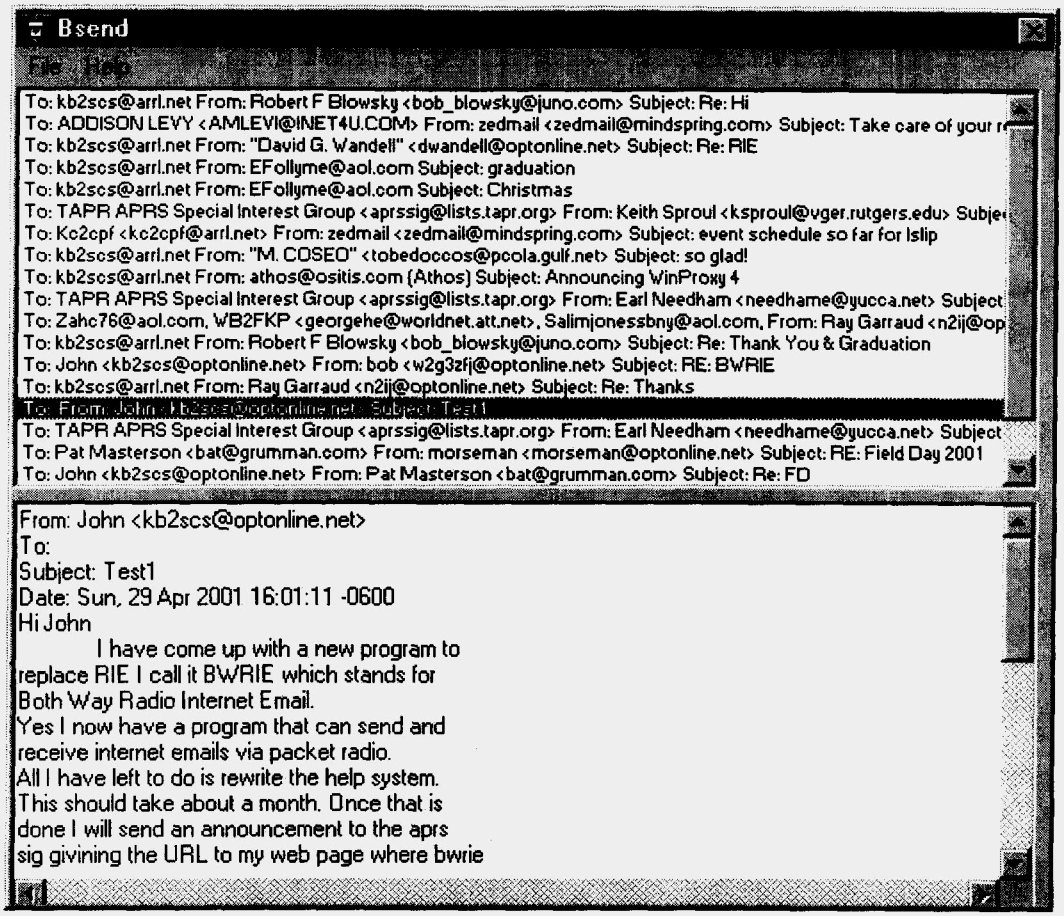

Bsend
File Help
To: kb2scs@arrl.net From: "David G. Wandell" <dwandell@optonline.net> Subject: Re: RIE
To: kb2scs@arrl.net From: EFollyme@aol.com Subject: graduation
To: kb2scs@arrl.net From: EFollyme@aol.com Subject: Christmas
To: TAPR APRS Special Interest Group <aprssig@lists.tapr.org> From: Keith Sproul <ksproul@vger.rutgers.edu> Subje
To: Kc2cpf <kc2cpf@arrl.net> From: zedmail <zedmail@mindspring.com> Subject: event schedule so far for Islip
To: kb2scs@arrl.net From: "M. COSEO" <tobedoccos@pcola.gulf.net> Subject: so glad!
To: kb2scs@arrl.net From: athos@ositis.com (Athos) Subject: Announcing WinProxy 4
To: TAPR APRS Special Interest Group <aprssig@lists.tapr.org> From: Earl Needham <needhame@yucca.net> Subject
To: Zahc76@aol.com, WB2FKP <georgehe@worldnet.att.net>, Salimjonessbny@aol.com, From: Ray Garraud <n2ij@op
To: kb2scs@arrl.net From: Robert F Blowsky <bob_blowsky@juno.com> Subject: Re: Thank You & Graduation
To: John <kb2scs@optonline.net> From: bob <w2g3zfj@optonline.net> Subject: RE: BWRIE
To: kb2scs@arrl.net From: Ray Garraud <n2ij@optonline.net> Subject: Re: Thanks
To: From: John <kb2scs@optonline.net> Subject: Test1
To: TAPR APRS Special Interest Group <aprssig@lists.tapr.org> From: Earl Needham <needhame@yucca.net> Subject
To: Pat Masterson <bat@grumman.com> From: morseman <morseman@optonline.net> Subject: RE: Field Day 2001
To: John <kb2scs@optonline.net> From: Pat Masterson <bat@grumman.com> Subject: Re: FD
To: TAPR APRS Special Interest Group <aprssig@lists.tapr.org> From: Stan Horzepa <stanzepa@mail2.nai.net> Subjec
Finished

Figure 4


Bsend
File Help
To: kb2scs@arrl.net From: Robert F Blowsky <bob_blowsky@juno.com> Subject: Re: Hi
To: ADDISON LEVY <AMLEVI@INET4U.COM> From: zedmail <zedmail@mindspring.com> Subject: Take care of your
To: kb2scs@arrl.net From: "David G. Wandell" <dwandell@optonline.net> Subject: Re: RIE
To: kb2scs@arrl.net From: EFollyme@aol.com Subject: graduation
To: kb2scs@arrl.net From: EFollyme@aol.com Subject: Christmas
To: TAPR APRS Special Interest Group <aprssig@lists.tapr.org> From: Keith Sproul <ksproul@vger.rutgers.edu> Subje
To: Kc2cpf <kc2cpf@arrl.net> From: zedmail <zedmail@mindspring.com> Subject: event schedule so far for Islip
To: kb2scs@arrl.net From: "M. COSEO" <tobedoccos@pcola.gulf.net> Subject: so glad!
To: kb2scs@arrl.net From: athos@ositis.com (Athos) Subject: Announcing WinProxy 4
To: TAPR APRS Special Interest Group <aprssig@lists.tapr.org> From: Earl Needham <needhame@yucca.net> Subject
To: Zahc76@aol.com, WB2FKP <georgehe@worldnet.att.net>, Salimjonessbny@aol.com, From: Ray Garraud <n2ij@op
To: kb2scs@arrl.net From: Robert F Blowsky <bob_blowsky@juno.com> Subject: Re: Thank You & Graduation
To: John <kb2scs@optonline.net> From: bob <w2g3zfj@optonline.net> Subject: RE: BWRIE
To: kb2scs@arrl.net From: Ray Garraud <n2ij@optonline.net> Subject: Re: Thanks
To: From: John <kb2scs@optonline.net> Subject: Test1
To: TAPR APRS Special Interest Group <aprssig@lists.tapr.org> From: Earl Needham <needhame@yucca.net> Subject
To: Pat Masterson <bat@grumman.com> From: morseman <morseman@optonline.net> Subject: RE: Field Day 2001
To: John <kb2scs@optonline.net> From: Pat Masterson <bat@grumman.com> Subject: Re: FD
From: John <kb2scs@optonline.net>
To:
Subject: Test1
Date: Sun, 29 Apr 2001 16:01:11 -0600
Hi John
I have come up with a new program to
replace RIE I call it BWRIE which stands for
Both Way Radio Internet Email.
Yes I now have a program that can send and
receive internet emails via packet radio.
All I have left to do is rewrite the help system.
This should take about a month. Once that is
done I will send an announcement to the aprs
sig givining the URL to my web page where bwrie

Revisiting the TNC firmware

Joachim Elen, ON1DDS
Unie der Belgische Amateurzenders – Radioclub Sint-Truiden (UBA-RST)
Naamsesteenweg 333, B3800 Sint-Truiden, België
e-mail: on1dds@qsl.net

Keywords
TNC, AX.25, SLIP, PPP

Abstract

This paper describes the implementation of a bridge in a TNC. By letting a TNC make the translation between AX.25 and a popular link layer protocol, countless new possibilities arise. Supporting SLIP or PPP would allow us to transparently attach our TNC to any device with a serial port, from personal computer to mobile handset. Users without specialized knowledge can start using complex network protocols like TCP/IP over radiofrequencies as they do on the Internet. This way, new networking technologies can be adopted or developed by radio amateurs.

Introduction

Except in gateways to the Internet, our European packet network generally does not support an advanced network protocol. Today, in 2001, we're still maintaining bulletin board systems. By nature, BBS's are driven manually, eliminating the necessity for further research on high-speed packet radio.

Packet radio might revive through a transparent introduction of modern network protocols like TCP/IP, Novell IPX, AppleTalk and others. SV2AGW[1] demonstrated this principle with his software. In this paper another approach is demonstrated.

Implementation

One major problem we have always been confronted with was the lack of hardware compatibility with any industrial communication standard. If we want to take advantage of the evolution in the commercial world, we have to commit ourselves to these standards and provide our users a 100% compatibility with them. We can revalue our TNC by making it work as a telephone modem and maintain AX.25 as our link protocol. The TNC can operate as a bridge - a device to translate AX.25 to another link protocol like PPP[2] or SLIP[3]. This would make the TNC a transparent device in either direction. Only this way we can guarantee (AX.25) net-connectivity for handheld devices and operating systems that don't support AX.25. This requires new developments on TNC firmware.

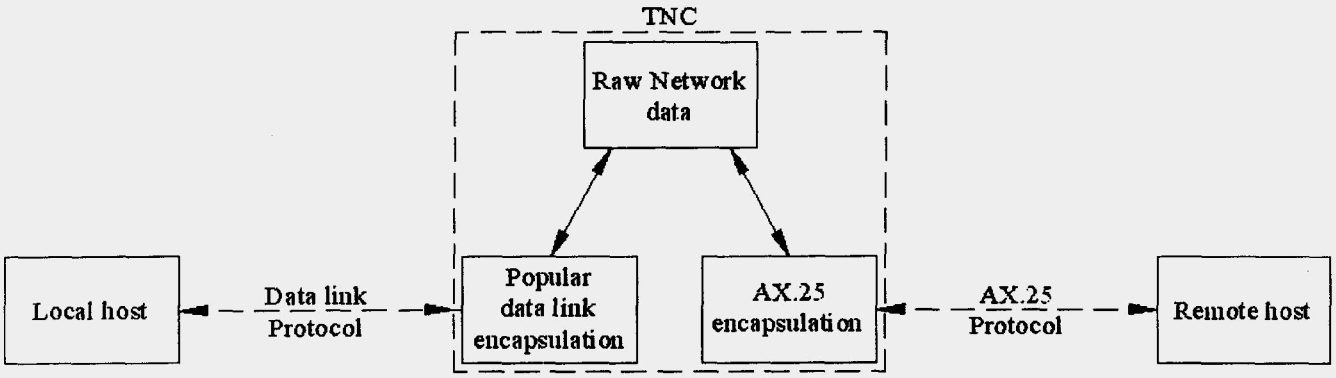
Command mode

Traditional modems generally use AT commands and S-registers. The same standard can be used for amateur radio purposes. Amateur specific parameters can be easily implemented as extended AT

instructions and the S-registers can hold AX.25 specific flags and parameters. To go ONLINE, the traditional telephone number can be altered into the destination call sign with optional via's.

ONLINE mode

Once online, the TNC will communicate PPP or SLIP with the computer and AX.25 on the frequency. The bridge functionality that takes care of the translation between both the protocols is easy to understand and is depicted in the following diagram:



The AX.25 frame exchange itself can be accomplished in either VC or UI mode, depending on the personal preferences of the one implementing the functionality. Each mode has its drawbacks and advantages.

AX.25 Unnumbered information

The implementation of UI is relatively easy. A UI connection relies on a link quality better than 90%. Since missing frames have to be detected by a higher level time-out mechanism. By default, popular operating systems implement these timers with exponentially growing retry times (often not providing configuration options. Baud rates of less than 4800 baud with busy frequencies affect both these timers and the link quality (collisions). This leads to infinite delay times and timed out applications. Nevertheless, a good quality and a higher baud rate overcomes this problem and can guarantee acceptable data transmission.

Virtual circuit

The implementation of VC is much more complex. A VC connection does not demand links of such quality because AX.25 provides a reliable connection autonomously. The drawbacks are similar to the ones in UI mode. Not only low baud rates and busy frequencies, but also the transmission of acknowledge frames affects the higher level timers. Additionally, AX.25 forces retransmission of lost frames – again requiring bandwidth. These retransmissions inavoidably cause a distorted round-trip time calculation, ending up in a unnecessary datagram retransmission. On the other hand, if the TNC could discard datagram duplicates, this weird behavior could be better controlled. Usage of VJ[4] compressed TCP frames decreases bandwidth tremendously. I have tried to implement VJ compression in a UI environment, but it was useless because the link quality has to approach the 100%.

Routing

A telephone modem is limited to providing a single point-to-point connection. A TNC however is a point-to-multipoint device. One might think that the nature of SLIP or PPP prevents the TNC from making multipoint connections. This is not true. Keeping a limited ARP table – either static or dynamic – in TNC memory and directing each outgoing frame depending on its ARP resolution, allows us to set up a direct link to each host within reach. Networking systems like TCP/IP however compel

client/server types of operation. Therefore, routing systems are only required when next to the default server one wants to connect a colleague radio amateur running services on its host or when multiple servers can be reached simultaneously.

MCB152[6] – an implementation

To check the true value of our philosophy, Walter Machiels, ON4AWM constructed the MCB152 TNC. This is a modular TNC, initially built for test purposes. The MCB152 is an 80c152jb based microcontroller board with 64 Kbytes of CODE memory and 64 Kbytes of DATA memory. The 80c152jb is an 8031HB with integrated SCC (Serial Communication Channel). A Baycom[5] USCC modem is to be plugged on the MCB152 and connected to the transceiver. The MCB152 is designed as a development board and has an EPROM containing a firmware loader. The upload is done using a copy command to the computer's serial port, but might, alternatively, be burned in EPROM. The Firmware was written by ON1DDS.

The current SLIP firmware implements an AT command interpreter with additional commands for packet radio purposes. The string in the following example sets TX-delay=15, slot time=7, TX-tail=0, persistence=63 and source call sign= "ON1DDS".
AT &TXD=15 &SLOT=7 &TXT=0 &P=63 C"ON1DDS"

To go online, one can use a command like AT DT ON0BAF.10@ON0EUL

The current firmware version processes AX.25 UI frames only, but a full AX.25 v2.2 implementation is under development. We currently use a 9600 baud DK9RR-FSK modem, but tests with a 153600 baud DF9IC/DG3RBU-FSK modem (at the highest baud rate) have proved to be functioning as well.

We have tested the MCB152 on most Microsoft Windows versions, Mac OS and Linux. None of them caused problems. Installation of a standard modem driver, normally delivered with the OS, is the only requirement. Dial in and ... enjoy TCP/IP packet radio using your favourite Internet software.

Unfortunately its not that simple, because this way of working requires an up and running TCP/IP server which, to prevent TCP/IP users from being isolated from the rest of the packet network must provide data exchange with other AX.25 users. We could use a NOS-like environment, but that would not be in line with our way of thinking – being as compatible as possible with the industrial standards. Therefore, Gert Leunen, ON1BLU has set up a TCP/IP server[7], running native Linux. A report on this project "TCP/IP and radio amateurism. - A UBA-RST TCP/IP Taskforce project" is presented elsewhere in this proceeding.

For backward compatibility, the AT @K command is implemented. This command switches the TNC to KISS mode. SV2AGW added this initialization string for the MCB152 in his software as well.

Conclusion

Although SLIP and PPP TNCs are no longer compatible with the traditional AX.25 in text mode, the advantages are obvious. Immediate compatibility with traditional telephone modems puts our TNC back in line with current Internet devices. Radio amateurs should be provided with the same environment as they are used to at home and at work. This is true for both hobbyists which prefer to surf the packet network for information as for professionals setting up a server. Hobbyists can focus on application

programming, without worrying about compatibility issues any longer. Once again there is a need for developing high speed modems and dedicated transceivers. The need for good servers arises. These servers are 100% compatible with Internet servers, a good reason for young people to become radio amateurs. They can do tests that will not be allowed by providers on the real Internet and learn about upcoming technologies during their spare time. And last but not least, average packet radio users can take advantage of those developments without having to worry about the technical background.

UBA-RST TCP/IP TaskForce:

- **Gert Leunen (ON1BLU)** - *TCP/IP server setup, Belgian IP/DNS coordination*
  Homepage: http://www.qsl.net/on1blu
  AMPR-mail: on1blu@on0baf.baf.be.ampr.org
  Internet E-mail: on1blu@qsl.net
- **Joachim Elen (ON1DDS)** - *Firmware development*
  Homepage: http://www.8052.com/users/joachim.elen
  AMPR-mail: on1dds@on0baf.baf.be.ampr.org
  Internet E-mail: on1dds@qsl.net
- **Walter Machiels (ON4AWM)** - *Hardware development*
  Homepage: http://home.worldonline.be/~vda10786)
  AMPR-mail: on4awm@on0baf.baf.be.ampr.org
  Internet E-mail: walter.machiels@pandora.be

References

[1] ROSSOPOULOS, G. SV2AGW
homepage: http://www.raag.org/sv2agw

[2] PERKINS, D. *Point-to-Point Protocol: A proposal for multi-protocol transmission of datagrams over point-to-point links*. ARPANET Working Group Requests for Comments, DDN Network Information center, SRI International, Menlo Park, CA, Nov. 1989. RFC-1134

[3] ROMKEY, J. *A nonstandard for transmission of IP datagrams over serial lines: SLIP*. ARPANET Working Group Requests for Comments, DDN Network Information Center, SRI International, Menlo Park, CA, June 1988. RFC-1055

[4] V. JACOBSON. *Compressing TCP/IP headers for low-speed serial links*. ARPANET Working Group Request for Comments, Real Time Systems Group, Lawrence Berkeley Laboratory, Berkeley, CA, Feb. 1990. RFC-1144.

[5] BAYCOM, Bavarian Packet Radio Group
homepage: http://www.baycom.org

[6] The MCB152 project
homepage: http://www.caseconsole.com/mcb152

[7] LEUNEN, G. ON1BLU. *The linux TCP/IP server configuration page*
http://www.qsl.net/on1blu – Linux server section

HF Digital Voice Transmission using an OFDM Modem with Space-Time Coding

Matt Ettus, N2MJI
matt@ettus.com
http://ettus.com

August 7th, 2001

Abstract

The High-frequency (HF) radio channel presents a unique challenge to the modem designer. It is characterized by large delay-spreads, rapid fading, and impulse noise. Those modems which have been successful in this environment have relied heavily on time-diversity (through coding and interleaving), or very low bit rates. However, long interleaver delays are not tolerable in a two-way voice contact, and so some other means of improving reliability is necessary if digital voice traffic is to be accomodated.

This paper discusses the design of a digital voice HF modem which uses orthogonal frequency division multiplexing (OFDM), some times referred to as a "parallel-tone" modem. While OFDM provides some frequency diversity, there is no inherent time diversity. In order to make up for this, spatial diversity is used, both on the transmit and on the receive sides.

1 Introduction

This project was both inspired and greatly influenced by the work of Charles Brain, G4GUO[1]. The goal of this project is to design a modem for the amateur HF bands which is capable of transmitting digital voice of a quality equal to, or greater than that of standard SSB. The aim is to minimize the amount of special hardware necessary, and so standard PCs and sound cards are used for signal processing and acquisition. A basic system will only need a standard HF radio with good frequency accuracy. A system capable of diversity transmission or reception would require a second antenna and a second radio system.

The modem uses OFDM modulation with DQPSK(differential quadrature phase shift keying). The occupied bandwidth is the full 2.7 kHz of the amateur HF voice channel. It sends a net of 2400 bits per second (4600 with coding), and will be able to take advantage of transmit diversity, receive diversity, or both, if available. A 2400bps vocoder is used to compress the voice down to the data rate of the modem.

2 Vocoder

The first obstacle to the transmission of digital voice over HF is the very low bit rates which can be sent. It was decided that 2400 bits per second was the highest rate that could be expected to work in moderate conditions. Unfortunately, this severely constrains the choice of vocoder, as most low-rate vocoders are proprietary.

The first, and most obvious choice for a vocoder is the LPC-10e standard, used by the U.S. military, among others. It is not patent encumbered, and public domain implementations are available. It was developed in the early 1980's, and unfortunately, its quality is not great. It is quite usable, however.

In the last few years, several 2400 bps vocoders were created, mainly to compete to be the official replacement for LPC-10e (with the goal of better sound quality than 4800 bps CELP). Among those are AMBE, from DVS Inc, used in [1], and the winning entry, MELP2400. MELP, as the winner, is heavily documented[5], and source code is available. It is not "free" in that it is owned by a corporation which licenses it[1]. It is not certain in what way this would impact its use in amateur radio, but for experimentation purposes that is not a huge concern. There is always the option of falling back to LPC-10e if there are significant entanglements.

3 Modem Design

3.1 Modulation

In OFDM, many carriers are modulated at a low rate. Conventionally, this would be done with many oscillators, but in modern systems the inverse fourier transform is used. The magnitudes and phases of each carrier are written into the appropriate bin, and the IFFT creates the corresponding time-domain signal which is transmitted. On the receiving side, a fourier transform is performed, and the magnitudes and phases of the transmitted carriers are recovered. Precise alignment of frequency and timing for the fourier transform window is required, or orthogonality is lost, and self-interference occurs.

One advantage of OFDM is that since the symbols are modulated at a low rate, the sidebands do not extend far beyond the edges of the signal. Because of this, pulse shaping (i.e. root raised cosine filtering) is usually not used. Also, since the individual tones are spaced at the minimum orthogonal width, greater spectrum efficiency can be obtained. In the limit of many tones, BPSK gives one bit per second per hertz, QPSK achieves two, 8PSK three, etc.

In channels with multipath, the last part of the previous symbol will interfere with the beginning of the next one. In order to combat this, guard times were originally used between symbols. It was later shown that by cyclically extending the symbol, and ignoring the beginning (where energy from both symbols is present), orthogonality could be maintained.[2] This is known as the cyclic prefix (CP). It is formed by copying the last few samples to the beginning of the transmitted symbol.

In this modem, we will be sending 2400 bits per second, encoded at rate 12/23. Thus, we have to send 4600 bits per second. We will be using DQPSK, which gives us 2 bits per carrier per symbol. The carriers are spaced at the inverse of the symbol rate,

[1] ASPI, Inc. claims unspecified "Intellectual Property" over MELP. Use at your own risk.

[2] Note that this does not remove the effects of multipath. Each frequency bin will undergo independent flat fading. They just won't interfere with each other.

which we choose to be equal to the fame rate of the vocoder, 22.5 ms. Each frame contains 54 bits. We use a cyclic prefix of 2.5 ms, leaving 20ms for the useful symbol time. The tones need to be spaced to be orthogonal over 20ms (i.e. 50 Hz), not 22.5ms (44.44 Hz), which costs us some spectrum efficiency.

One of the main disadvantages of OFDM modulation is the very high peak to average power ratio (PAR). Since most power amplifiers are limited by peak power (and FCC regulations limit peak power), a bigger amplifier is necessary for the same average power as single-carrier systems. This power backoff also causes low power efficiency, since the amplifiers must be linear. Some methods for reducing the PAR of OFDM have been proposed, but the one currently implemented uses random initial phases for the first symbol.

3.2 Coding

In order to provide protection against noise, forward error correction (FEC) is used. The Golay (23,12) code was selected because a free implementation is available, and it is quite simple. The fact that the rate is slightly less than 1/2 (for each 12 bits in, 23 coded bits are generated) is also quite helpful to us, since it allows us to fit within standard radio passbands without having to move to a higher-order modulation like 8PSK. More advanced coding, utilizing convolutional codes and soft-decisions is under consideration.

3.3 Synchronization

Proper synchronization, both in symbol timing and frequency, are very important for successful OFDM reception. Small timing errors cause phase rotations proportional to frequency, and a loss of orthogonality, resulting in intercarrier interference (ICI). Larger timing errors result in intersymbol interference (ISI). Both cause significant bit error rate (BER) degradation. Frequency errors of just a few percent of the intercarrier spacing cause significant ICI because of the loss of orthogonality. In our modem, the intercarrier spacing is 50 Hz, so relying purely on the accuracy of the frequency reference is not practical.

We use a maximum likelihood joint estimator for both time and frequency offset which was proposed in [7]. This estimator takes advantage of the cyclic prefix (which would otherwise be wasted energy). By computing the correlation between samples separated by the frame size, the CP is detected. The relative phase of the correlation at that point gives the frequency offset.

This estimator is capable of coarse acquisition of timing, as well as fine tracking. For frequency, it is capable of fine tracking, to within better than 1% of the intercarrier spacing. Some other means of tracking frequency to within +/- one half of the intercarrier spacing is necessary. For now, we rely upon the frequency accuracy of the transmitter and receiver to get close enough for the ML estimator to work. This requires better than 1 ppm accuracy on 20 meters.

3.4 Spatial Diversity

Because the HF channel is time-varying, most HF modems use extensive interleaving. This ensures that long sequences of bits which are lost to deep fades will be spread out, allowing the error correction coding (FEC) to work. This does not work well with voice communications because of the long delays, sometimes as much as 10 seconds. Instead of time diversity, we must find another way of mitigating the

effects of fading – spatial diversity.

Spatial diversity takes advantage of what every operator intuitively knows – that when one spot undergoes a fade, moving the antenna can help. Instead of moving the antenna, however, we have multiple antennas, the signals from which we can combine coherently. Little has been published on the subject of coherence distance[3] on HF channels, but a spacing of 2 wavelengths should be sufficient. For those without that much space, polarization diversity has been shown to provide nearly as much gain.

For receive diversity, maximal-ratio receiver combining (MRRC) will be used. In MRRC, the received signals are added together after multiplication by the complex conjugate of the path gain. Thus, each signal is weighted by its received strength. This can be used with an arbitrary number of antennas. Of course, an accurate estimate of the [complex] path gain is necessary. The first two symbols at the beginning of each transmission can be used to get an initial estimate, which will need to be continually updated after subsequent symbols are received.

Transmit diversity is more complex. In order for the receiver to be able to separate the signals from each of the transmitters, they must be encoded differently. The method used in this project was first proposed in [2], and was applied to OFDM in [3]. It is limited to two transmit antennas. Basically, at time (or frequency in the OFDM case) n, transmitter T_0 sends signal s_0, and T_1 sends s_1. At time (or frequency) $n+1$, T_0 sends $-s_1^*$, and T_1 sends s_0^*. Two slot are used to send two symbols, thus the same data rate is maintained.

Assuming h_x is the [estimated] path gain from antenna x to the receiver, and r_x is the received signal at time (or frequency) x, then the receiver need only perform the following computations to decode the signal:

$$s_0 = h^*_0 r_0 + h_1 r^*_1$$
$$s_1 = h^*_1 r_0 - h_0 r^*_1$$

The math gets a little more complicated with multiple receive antennas, but the concept is the same. Either the transmitter, the receiver, or both could utilize diversity in this system. A useful result of this is that if only one participant has a dual-antenna setup, both sides can still take advantage of the 2-way diversity. Of course, if both have dual-antenna setups, then 4-way diversity is possible, with a corresponding increase in diversity (there are now 4 paths between transmitters and receivers).

[3] Coherence distance is the minimum spacing between antennas which provides essentially uncorrelated fading.

4 Future Work

Currently, most of the basic work is completed. OFDM transmission and reception work, as do the time and fine-frequency synchronization. An algorithm for estimating coarse frequency error will be necessary if high-accuracy frequency references won't be used in the transceivers.

In order to implement the receive processing which is required for the transmit and receive diversity schemes, a channel estimator will be necessary. Several options for this are being investigated, but it is unclear how the very short training sequences will affect performance.

Hardware to implement the diversity scheme will also be required. For the prototypes, T2 transmitters and R2 receivers [6] will be used, due to their simplicity, low cost, high dynamic range, and flat passband.

The various methods of systematically reducing PAR and linearity requirements will be investigated.

A coding scheme which has more coding gain than Golay codes, and which can use soft decision information will likely be added. It may be necessary to switch from DPSK to 8PSK with trellis-coded modulation if typical radio passbands are not clean enough. In this case, an outer Reed-Solomon code could easily be added, possibly making up for the loss of power efficiency which comes with 8PSK.

References

[1] Charles Brain, "A practical approach to implementing H.F. digital voice in the amateur service," *18th ARRL and TAPR Digital Communications Conference*, pp. 17-21, September 1999. See also `http://www.chbrain.dircon.co.uk/dvhf.html`

[2] Sivash M. Alamouti, "A simple transmitter diversity scheme for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 3, pp. 451-460, March 1999.

[3] K. F. Lee and D. B. Williams, "A space-frequency block-coded OFDM transmitter diversity technique," *IEEE GLOBECOM 2000*, San Francisco, CA, November 2000.

[4] Martin C. Gill, "Coded-waveform design for high speed data transfer over high frequency radio channels," Ph. D. Dissertation, University of South Australia, February 1998.

[5] See `http://www.plh.af.mil/ddvpc/melp.htm` and `http://www.aspi.com/products/speech/melp.html`

[6] Rick Campbell, "High-performance, single-signal direct-conversion receivers," QST, January 1993.

[7] J.-J. van de Beek, M. Sandell, and P. O. Börjesson, "ML estimation of time and frequency offset in OFDM systems," *IEEE Transactions on Signal Processing*, vol. 45, no. 7, pp. 1800-1805, July 1997.

TCP/IP and radio amateurism
A UBA-RST TCP/IP TaskForce project

Gert Leunen, ON1BLU
Unie der Belgische Amateurzenders (UBA) – Radioclub Sint-Truiden; EURO><LINK
Leeuwerweg 34, B-3800 Sint-Truiden, Belgium
on1blu@qsl.net

Abstract

Although several reports on TCP/IP projects have already been published, we felt our specific approach and vision could still figure as a contribution to the subject. The approach we will present here is one that addresses nearly all aspects of the network infrastructure (from hardware solutions, through network topology, up to services), focuses on transparency to the user and provides INTERACTION with legacy – as opposed to PORTING legacy into TCP/IP.

Introduction

Ever since Packet Radio saw its daylight during the eighties – with speeds up to 300 and 1200 baud's – it appears not to have evolved to better operation. Sure, there are many initiatives, 9600 baud G3RUH being the most persistent, but none of them is sufficiently wide spread. It's almost as if Packet Radio is the most noticeable sign of what's currently happening to radio amateurism in general (in Belgium at least): a fade out. Many discussions have been held on the causes, Internet and GSM being the most frequently referenced. In my opinion, there are 2 causes to this:

- Even though we have a wide spanning Packet Radio network (and the Packet Radio network currently is the only way to connect ALL HAMs on a 'PERMANENT' basis), the pace of its evolution now results in an arcane technology that's hard to work with. This has a negative effect on sharing technical knowledge and experience, and therefore prohibits large based experiments.
- Since radio amateurism hasn't evolved as the Internet did, for example, there's no need to experiment with new technologies. Transceivers are already very small, further improving them seems obsolete: such improvements won't let us talk faster :-), nor let us type faster (while working with our interactive BBS systems, to name just one).

The UBA-RST team concluded this particular Packet Radio domain should undergo a major evolutionary step. From the very beginning, we had two goals in mind:

- HAM-operators who are not interested in computers or Packet Radio itself - but want to use these TOOLs to exchange their experiences, should be able to do this in a very simple way, preferably using software they're already accustomed with (in other words: without having to learn various software packages).
- HAMs who feel to perform technical experiments should be provided a platform where they can learn things that are relevant to them. Let me illustrate: the Packet Radio protocol - AX.25 - is specifically designed for Amateur Radio. If technicians are forced to develop directly upon this AX.25 platform, they can hardly use their experience anywhere else. Ultimately, this means they must start producing their own goals, often not providing any contribution to radio amateurism in general.

So, now we come to the point why we choose for TCP/IP specifically:

- TCP/IP is the protocol that's currently being used on the Internet. If we superpose this protocol set on top of the bare AX.25 link protocol, we enable HAMs to use their favorite and accustomed Internet software (like Netscape Communicator, Microsoft Internet Explorer, Microsoft Outlook, etc). This would greatly simplify the usage of Packet Radio: not only by the related software being user-friendly itself, but also eliminating the need for learning additional software packages (the network and its services no longer dictate the client software to be used to exploit all its features – c.f. BBSs and terminal software supporting either YAPP or SPBIN protocols for binary transfers).
- TCP/IP is not only used on the Internet: even the smallest companies are switching to TCP/IP for use on their local networks (LAN). This means TCP/IP technology, TCP/IP knowledge and TCP/IP experience is as relevant as you can get! If you learned something on the AX.25 + TCP/IP network, you can apply this knowledge equally well in your professional life.

Especially the latter is very important, since it revives the "raison d'être" of radio amateurism. If one wants to perform real-life experiments on server technologies, he can only do this on the Amateur Radio network! Internet providers, companies and even universities WILL NOT allow you to experiment on their network (you could cause major damage when you suffocate their network and – as a consequence – potentially parts of the Internet as well). So, you could create an isolated network, but you can hardly call that a real-life test, can you?

What the project is all about

While we noticed many people were already experimenting with TCP/IP and high-speed Packet Radio, we never encountered a full-covering approach. So, even though our pure development efforts are somewhat limited (to the MCB-152 [1]), we generally collected many bits and pieces and tried to construct guidelines [2] for SysOps to bring their nodes back to life.

This was a step by step process as we'll illustrate shortly, but on realizing each individual step we considered one aspect thoroughly: LEGACY. None of our realizations would ever be accepted if they ruled out a group of users. So we must be compatible with existing Packet Radio systems (as far as modulation and coding is concerned), allow Packet Radio users to pass the system, allow mails and bulletins to be exchanged between TCP/IP users and BBS users, etc.

For the remainder of this text, we'll talk about AX.25 users and TCP/IP users. Of course, TCP/IP users also run AX.25, but they don't use the AX.25 protocol **consciously**.

Table 1 - The OSI reference model

Application layer
Presentation layer
Session layer
Transport layer
Network layer
Data-link layer
Physical layer

Generally, Packet Radio was never meant to be the ultimate communication platform itself. If you take a look at the Digital Communications section of the well-known ARRL handbook [3] (even for an eighties-edition), you'll notice the OSI layered communication model (Table 1). While TCP/IP consists of lesser layers, we'll use the OSI layers bottom-up to tell our story step-by-step or layer-by-layer.

Raw Communication

Let's start off with the physical layer. The physical layer is the grouping of communication media and the collection of equipment that's necessary for modulating and demodulating raw data on a specific

medium. For sound, the medium is the propagating vibration of molecules in a liquid or gas environment and human equipment are the vocal chords and ears. We, radio amateurs, are interested in the electromagnetic medium and the accompanying transceiver equipment, of course.

On our 9k6 user accesses at 70cm, the T7F transceiver [4] – yielding very good results – is typically used. For high speed user access, we're evaluating the 23cm data transceiver [5], which should allow speeds up to nearly 200k baud.

For modulation and demodulation, we typically use BayCom [6] USCC-type modems (DK9RR and DF9IC modems). They can be plugged into our node's USCC card and onto our MCB-152 (as you'll see in the next section). The high speed user access tests are performed using BayCom's prototype high-speed DF9IC/DG3RBU modem.

We're experimenting with a repeater-like system (Figure 1), separating user-access uplink and downlink frequencies. This repeater-like system will perform either audio-repeating or echo-duplexing and possibly keep its transmitter on air (reducing TX/RX switching, which takes a considerable portion of the time required to send frames at these speeds). The major advantage of this system is that clients that are unreachable in simplex are now able to detect the other's signal, thereby significantly reducing the number of collisions (since 'invisibility' is a major cause for repeated collisions on user inputs). The node/router/server itself could use this repeater-like system as the users do, or could tap from the receiver of this "repeater" and send modulated data directly into its transmitter.

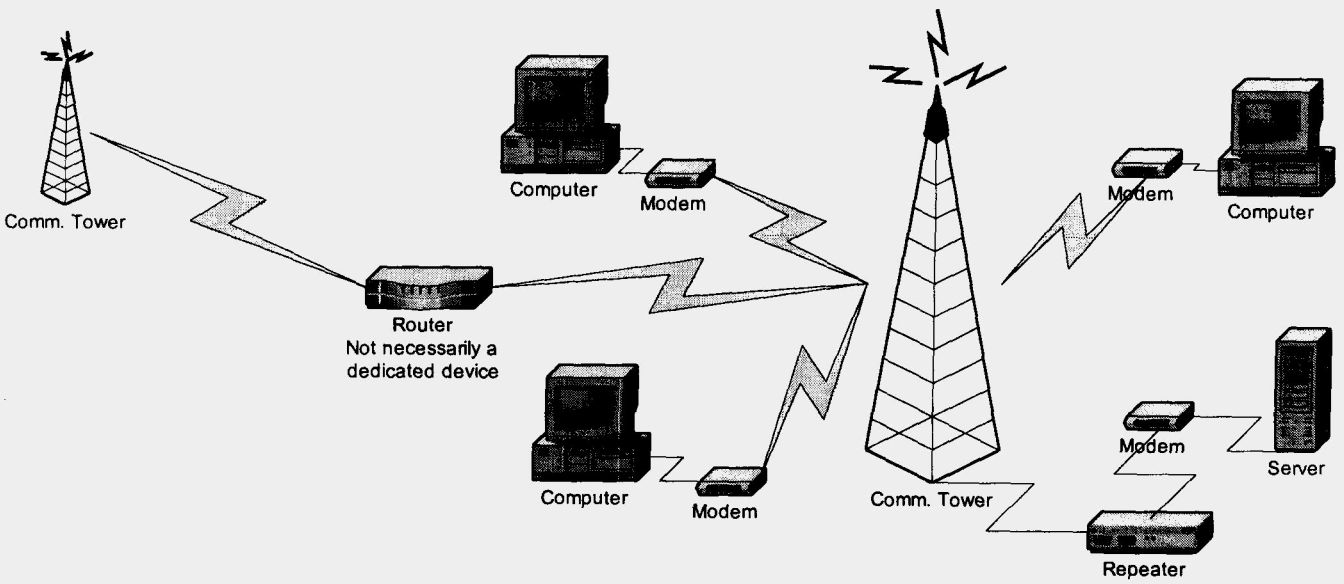


Figure 1: Repeater setup for high speed with better collision avoidance

If this repeater-like system succeeds, it could replace current expensive and dedicated links in some foreseeable future: if someone wants to start a new node, he only needs to buy one set of equipment (for linking up to the interregional repeater), instead of several sets for linking to several nodes. Interregional repeaters are then linked by nodes that volunteer linking to 2 interregional repeaters, essentially acting as routers at that time.

Generally, by adding an additional TCP/IP load to AX.25, we hope to create an essential need to further improve digital transmission performance.

Packet Radio

At the data-link layer, we find our AX.25 protocol. The data-link layer is actually the collection of software that provides "courtesy" protocols (keep silent while others are talking, for example). When a TNC device is used, AX.25 is handled as part of that TNC's firmware (next to command handling).

Our contribution to the link layer is the development of the MCB-152, which was mentioned earlier. These were our design goals when developing the MCB-152:

- it should support high speed communication evolution, without needing to buy entirely new equipment,
- for initial appreciation it must be compatible with current TNCs (1200 and 9k6 G3RUH-compatible) and current software (KISS firmware), and
- it should require minimal configuration effort.

The first two goals are realized by separating the controller hardware from the modem itself through a connection header that's pin-compatible with BayCom's USCC-type modems and by developing KISS firmware.

The third goal was realized by developing our famous SLIP firmware. After firmware download, the MCB-152 will act as an ordinary telephony modem (interpreting the highly standardized AT Hayes command set). The user dials as 'telephone number' the link address (call-sign) of his local IP router and as soon as the dial is complete, the MCB-152 will turn into a SLIP-to-AX.25 translator (and vice versa). So, the user only needs to setup an additional Internet account, using one of the generic modem drivers that are provided with Operating Systems like Microsoft Windows '95/'98/Me/NT/2000, etc. The MCB-152 also auto-detects the baud rate used on its serial port.

At the time of this writing, Joachim (ON1DDS) is re-developing the firmware for better performance, incorporating AX.25 V2.2 support.

One last word on our MCB-152: upon developing the firmware, we noticed the lack of supporting libraries for programming the Intel 80c152 chip. At that point, we decided not to limit the MCB-152 for TNC usage. Instead, Joachim (ON1DDS) developed a complete development library and collected all freely available development environments and documentation on a single CD. This enables the MCB-152 to be used for education, satellite tracking, you name it!

While the MCB-152 provides one solution to bridge the gap between popular Operating Systems and TNCs (by simulating commercial landline modems), SV2AGW [7], FlexNet [8] and Linux adopt another solution: they provide drivers so the OS handles the TNC as just another LAN Network Interface Card. While we like the elegancy of this solution, we noticed it's generally hard to keep up with OS evolution: SV2AGW and FlexNet supports Windows '95/'98/Me(?), only recently Windows NT/2000 and probably not (yet) Windows XP. For Linux, I recently got the impression the networking part of the 2.4 kernel has been considerably revised and I still have to figure out its effect on my next-generation server. That's why we thought there would be a reason for the existence of the MCB-152.

TCP/IP

It's only at the network layer that TCP/IP appears for the first time. You read this right: without AX.25, TCP/IP simply can not operate on our radio frequencies. The TCP/IP protocol set is available on nearly all computer architectures and operating systems.

For completeness:

- The network layer is the collection of software that handles routing. It's the network layer software that reduces our personal "network knowledge" to identifying our local IP router and identifying our destinations (no longer bothering about the local node of our destination and possibly several intermediate routes). The network layer software should also redirect data appropriately (and transparently!) when intermediate nodes fail.

  When using the TCP/IP protocol set, IP (the Internet Protocol) is the routing mechanism used. It's a hop-to-hop routing protocol, meaning that routes are not calculated by the IP software on your machine. Instead, each node (including your machine) decides - for each individual packet - what neighboring node he will forward the packet to.

- The transport layer is responsible for transporting information. Until now, we only talked about packets of data; transport layer software will maintain an information stream on top of an unreliable network. Two kinds of transports are currently common: simple, unacknowledged datagrams (UDP protocol) and acknowledged in-order no-duplicate information transfer (TCP protocol).

In Belgium, we had to do some preparation, however. IP addresses were distributed at a too coarse level (provinces), and those provinces were assigned decimal coded identifiers. For as soon as a second TCP/IP server appeared in a single province, the problems started. Within a province, IP addresses were distributed sequentially. When several servers appeared, users typically operated through their nearest TCP/IP server. However, the IP address distribution didn't allow any binary mask to distinguish the users of different servers. Within a province, explicit host routes had to be created on all servers within that province, for all users in that province, resulting in large routing tables to be maintained manually.

Our TCP/IP TaskForce submitted a proposal [9] for introducing TCP/IP regions (corresponding to IP router availability). This allows for very simple network route entries. These regions are authoritative now, meaning that each region coordinator can distribute IP addresses freely within his assigned range of IP addresses. For now, region coordinators forward their assignments to the national IP coordinator, such that the latter can synchronize towards the worldwide ampr.org file.

In a near future, we hope **name server** (NS) entries **may be added to this** worldwide **ampr.org file**, thereby fully enabling the authority of the regions. Domain zones have already been introduced, however: my FQDN, for example, is on1blu.baf.be.ampr.org (even though there's a CNAME entry for on1blu.ampr.org pointing to this 'regional' FQDN). The approach where all names are assigned to the 'root' ampr.org domain was sufficient in the (x)NOS days (which had no DNS 'system', only individual entries), but conflict with original DNS concepts as they are adopted in professional DNS servers (like BIND). Name server entries would also enable us to experiment with mechanisms such as DHCP/DDNS.

Sessions and presentation

Currently, the TCP/IP layer model doesn't provide explicit session (some applications use cookies to simulate session management) and presentation layers (some applications implement MIME extensions), so we'll skip it for now.

Applications and services

So, we finally reached the application layer. The application layer is the collection of all protocol API libraries (HTTP, FTP, SMTP, etc.), used by client and server software.

User applications (like Netscape Communicator, Microsoft Chat, etc.) use these API libraries for information exchange across the network. What this all means is that a programmer doesn't need to program a script engine for automating routes to distant BBSs, downloading and uploading mails and bulletins, or any other network issue for each application he wishes to create. He can now concentrate on one thing: providing a user-friendly interface.

These applications, however, communicate with service providers (Microsoft Outlook, for example, communicates with an SMTP server and a POP server). This brings us to our final effort: setting up TCP/IP servers.

We currently have 2 options:

- a Windows NT based server and
- a UNIX based server (like Linux)

The first isn't quite an alternative, especially in the NT4 era. NT4 is a desktop operating system with many networking tools. Next to the fact that NT servers are generally too expensive for amateur radio servers, many services need to be bought separately as well (all of them being equally expensive). Furthermore, a driver should be developed to let NT communicate with our TNCs, somewhat like SV2AGW and FlexNet32 do for Windows '95/'98, as SLIP can't be used on our servers (since it can only carry IP frames and no native Packet Radio traffic, breaking our support for legacy). A PPP firmware could solve this, however.

Windows '95/'98 is not feasible for providing services. Windows 2000, however, is a far better environment, since it really is a network operating system and it finally complies with standards. Most Internet services are included now too, but it's still expensive and requires driver development.

We choose for Linux since

- it's free,
- it supports AX.25 in its kernel NOW and
- it comes with all – currently well-known – services, all for free as well.

The Linux server[1]

Our project specifically collected configuration information for ALL those services. We succeeded in providing our users with a wealth of services:

- E-mail service, through the sendmail SMTP-server and the IMAP/POP3 server from the University of Washington
- News bulletin service, through the INND NNTP-server
- Web browser service through the Apache web (HTTP) server. We also allow our users to put their personal home pages on our web server
- FTP, NFS and SMB (c.f. Network neighborhood in Windows operating systems) file services, etc.
- Remote login (telnet) service
- Chat/conference service through Undernet IRC server

[1] Providing individual references that apply to this section would bring us to far. Please check the 'Related links' section on my website [2].

- Domain name resolving through the BIND DNS server

Most of the server software mentioned is also very popular on the Internet for use by ISPs, universities, etc. Each of these services will facilitate information exchange enormously as schematics and source code can be mailed using drag & drop, project status can be published through web pages, etc.

Remember our permanent concern for legacy:

- We allow AX.25 users to participate in the IRC network.
- We also provide an Internet-type DX-cluster service that can be connected from both AX.25 and TCP/IP users (and which links up to the existing DX-cluster network): CLX.
- By using telnet, users are patched through to our node, so they can enter the legacy network (notice that you can't use legacy packet terminal software while you're dialed in on the same TNC, at least at the time of this writing for the MCB-152; PPP firmware and an appropriate 'TF TSR'-like hook would make this possible, however).
- And, extremely important, we provide an SMTP/NNTP <-> BBS gateway. Earlier, we ran a JNOS-box on our Linux server. This JNOS-box's TCP/IP stack was connected to the native Linux TCP/IP stack through a virtual SLIP link, allowing JNOS's SMTP and NNTP servers to exchange data with Linux's SMTP and NNTP servers. For forwarding, JNOS tapped Linux's AX.25 interface.

  Now, we use the mailgw and lnxforward packages. Mailgw is actually an MTA (Mail Transport Agent) and we configure sendmail and INND to exchange BBS-destined or BBS-source messages with an FBB-compatible system.

  Lnxforward is an FBB-compatible forwarding-only module (also providing a web interface for reading mails and bulletins). This allows for transparent exchange of mails and bulletins between BBS and TCP/IP users.

Instead of using the lnxforward package, we could also install yet another full-fledged FBB system on our server, but we prefer not to. By installing another classical BBS system, you don't send a consistent message that users would benefit from switching to TCP/IP. Instead, we closed a deal with a neighboring BBS SysOp by which we have a forward feed to our gateway and our local AX.25 users have a home BBS through our local node step up. In Belgium, we also standardized the mapping between BBS areas and NNTP news groups (c.f. [9]). Altogether, this is a clear illustration of what we mean when we talk about "INTERACTING with legacy as opposed to PORTING legacy".

Once the configuration of our server was completed, a 5-day seminar was organized (attended by many Belgian SysOps), a step-by-step installation and configuration guide [2] was completed, and all related configuration files (as used at our server) were published on the web. The project currently refers to Red Hat Linux 5.2 based systems, but work is underway to update the project to more recent distributions.

Conclusion

By introducing TCP/IP on our amateur frequencies, we brought radio amateurism back up-to-speed with industrial development/environment. As we noticed that major SQL (database) server builders like Sybase and Oracle make their latest versions freely available on Linux for development use and as we noticed free application servers – supporting Java and XML, which only recently exited hype stage in industry – for Linux, we are convinced radio amateurism can reconquer its pioneering role that characterized radio amateurism until a few decades ago.

To continue the illustration: for demonstrating TCP/IP at 76kbps, we issued voice-over-IP, which is still cutting-edge in industry (not that we have actual plans for applying this technology, but we did it!). And maybe this indicates the future of radio amateurism: at first, it was a technological hobby, then it partly became a communication hobby, but – mainly due to the Internet – it will return to its roots: technology through experiments.

Reference

Members of the UBA-RST TCP/IP TaskForce:

- Gert Leunen (ON1BLU)
  TCP/IP server setup
  Belgian IP/DNS coordinator - Network 44.144.0.0 (0xffff0000) - Zone be.ampr.org
  AMPR-mail: on1blu@on0baf.baf.be.ampr.org
  Internet E-mail: on1blu@qsl.net
- Joachim Elen (ON1DDS)
  Firmware development
  AMPR-mail: on1dds@on0baf.baf.be.ampr.org
  Internet E-mail: on1dds@qsl.net
- Walter Machiels (ON4AWM)
  Hardware development
  AMPR-mail: on4awm@on0baf.baf.be.ampr.org
  Internet E-mail: walter.machiels@pandora.be

[1] The MCB-152 project:
Walter Machiels, ON4AWM (hardware: http://home.worldonline.be/~vda10786)
Joachim Elen, ON1DDS (firmware: http://www.8052.com/users/joachim.elen)
Distribution: http://www.caseconsole.com/mcb152
Also read their paper "Revisiting the TNC firmware" in the proceedings of this conference edition

[2] The Linux TCP/IP server configuration project:
Gert Leunen, ON1BLU (http://www.qsl.net/on1blu, 'Linux server' section)
Gert Leunen, *Wireless Linux TCP/IP server seminar*, June 2000 (downloadable from website)

[3] The ARRL Handbook for Radio Amateurs, American Radio Relay League, yearly publication.

[4] Holger Eckardt, DF2FQ, *A synthesizer-controlled 70cm-transceiver for 9600bd packet radio*
DF2FQ@amsat.org
http://www.rlx.lu/~rl/t7f/HANDBUCH.HTM (German)
http://www.la3f.no/prosjekt/t7f/t7f_e.html (English)

[5] Bas de Jong, PE1JPD, *23 cm data-transceiver for high-speed packet*
http://www.dutch.nl/bdj

[6] BAYCOM, Bavarian Packet Radio Group
http://www.baycom.org

[7] George Rossopoulos, SV2AGW (http://www.raag.org/sv2agw)

[8] Gunter Jost, DK7WJ, *FlexNet* (http://www.afthd.tu-darmstadt.de/~flexnet)

[9] Gert Leunen, ON1BLU, *Organizational agreements for setting up a wireless TCP/IP network in Belgium (on top of AX.25)*, November 1999 (downloadable from website: http://www.qsl.net/on1blu, 'TCP/IP Taskforce' section, status track).

Digital Amateur TeleVision (D-ATV)

Thomas Sailer, HB9JNX/AE4WA, Wolf-Henning Rech, DF9IC/N1EOW,
Stefan Reimann, DG8FAC, Jens Geisler, DL8SDL

August 7, 2001

Abstract

In this article, we present a Digital Amateur TeleVision (D-ATV) transmission system. Its signal can be received by cheap set-top boxes available for approximately US$100. It offers a wide user-selectable trade-off between signal bandwidth and picture quality, and at 4.5 MHz −40 dB bandwidth achieves better picture quality than analog systems.

1 Introduction

Analog Amateur TeleVision (ATV) has been in service virtually unchanged for over 20 years. Now, the industry is rapidly moving to all digital systems. Therefore, the time has come also for Amateur TeleVision to move to digital systems.

In central Europe, a significant amount of spectrum has been reserved for Digital Amateur TeleVision (D-ATV) for a number of years already, but not much has happened so far. Therefore, we decided at the Packet Radio Conference in Darmstadt, Germany, April 2001 to build a system to be shown at the Friedrichshafen Convention in June 2001.

It is clear that a complex system like a digital TV system cannot be developed from scratch as a spare time project of by small group of people in such a short time. Therefore, we wanted to use as many commercially available modules as reasonably possible.

The most widely used digital TV system is called Digital Video Broadcasting (DVB) and is based on a family of standards pioneered by the European Telecommunications Standards Institute [3]. At its core is the MPEG2 audio and video compression standard [5]. ETSI further defined three different physical layers to accommodate different transmission media.

DVB-C DVB-Cable [7] has been designed for cable networks. It uses quadrature amplitude modulation (QAM) with large constellations. It requires highly linear transmitter and receiver amplifiers and is thus unsuited to the Amateur Radio environment.

DVB-S DVB-Satellite [6] was designed for the satellite channel and uses quadrature phase shift keying (QPSK). Being designed for nonlinear traveling wave tube (TWT) amplifiers, it has benign linearity

requirements. Power amplifiers can thus be built as Amateur Radio projects with reasonable complexity. Amateur TV enthusiasts tend to use directional antennas, which greatly reduce multipath propagation.

DVB-T DVB-Terrestrial [8] has been designed for the mobile terrestrial channel with heavy multipath propagation. It uses orthogonal frequency division multiplex (OFDM). Multiple or all transmitters of the same network may share a single frequency. Unfortunately, DVB-T requires complex baseband signal processing, and furthermore DVB-T set top boxes are still scarce.

We chose DVB-S, because a huge variety of set top boxes (satellite receivers) are available on the market, and their input range ($\approx 900MHz - 2GHz$) includes the 23cm amateur radio band. The receiver chain is simple to build. It consists of DVB satellite receiver, a mixer for bands other than 23cm, an antenna and a standard TV set. The transmit chain looks different. TV studio equipment can certainly be bought, but at a price tag that is well outside of what the typical experimenter can or want to spend. So it made sense to build our own transmit chain.

2 The D-ATV Transmitter

Figure 1 depicts a block diagram of the transmit chain. A standard analog video source such as a video camera sends the signal to the MPEG2 encoder. The MPEG2 encoder converts the signal into a digital format, compresses it and sends a MPEG transport stream multiplex data stream to the baseband processor. The baseband processor performs coding and modulation tasks and produces a baseband IQ signal. The up converter converts the baseband signal to the desired carrier frequency, which is then amplified by the PA and radiated at the antenna.

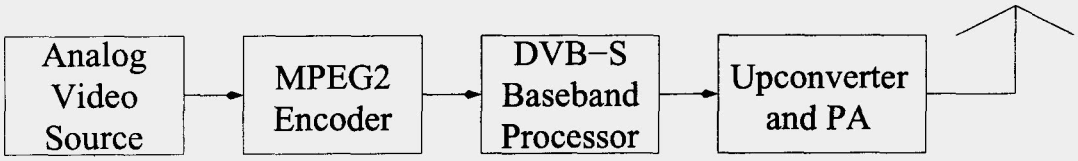


Figure 1: Block Diagram

2.1 The MPEG2 Encoder

Several companies are offering highly integrated MPEG2 encoder solutions. We chose the Fujitsu single chip MPEG-2 System Encoder LSI MB86390. An evaluation board from SR Systems [4] served our needs. Figure 2 shows the evaluation board.

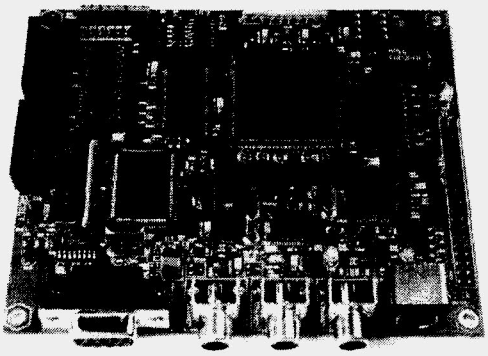


Figure 2: The Fujitsu MPEG2 Encoder Evaluation Board

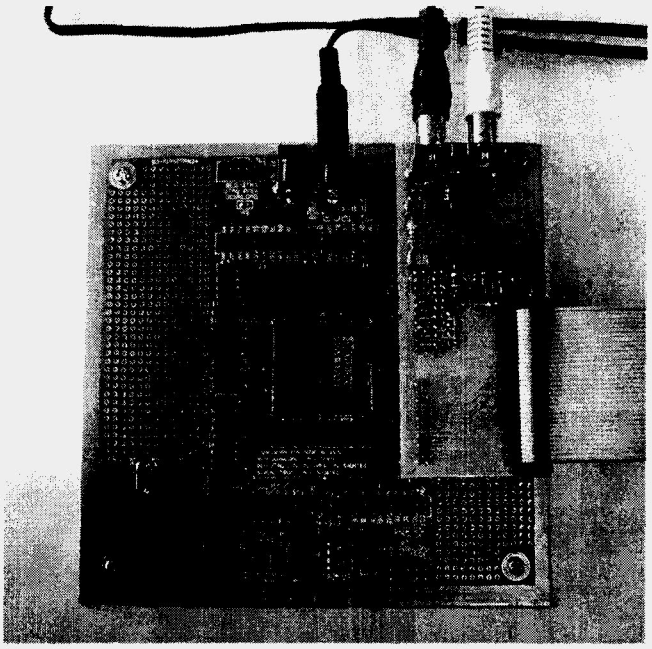


Figure 3: The Baseband Processor FPGA Board with D/A Converter Daughter Board

2.2 The Baseband Processor

Figure 4 depicts the functions performed by the DVB-S Baseband Processor. The first row of blocks operate on bytes, while the lower two blocks operate on bits.

First, the signal source, either the external MPEG2 encoder or the internal Test Data Source is selected by a jumper. The Test Data Source helped integration.

The Framing block extracts framing signals from the transport data stream and synchronizes the other modem blocks.

The Pseudo-Random Byte Sequence (PRBS) generator scrambles the data stream with a pseudo-random signal to ensure an adequate number of transitions in the signal.

The outer error correction encoder uses a Reed Solomon RS(255,239,8) code shortened to RS(204,188,8). The RS encoder operates in GF(2^8), that is on bytes. It takes an input block of 188 bytes (a MPEG2 transport stream packet) and adds 16 redundant bytes that help the receiver correct transmission errors.

The interleaver reorders the data stream. Its main purpose is to spread burst errors over multiple code words (that is 204 byte Reed-Solomon blocks).

The Parallel to Serial block converts the byte stream into a bit stream, which is then fed to the inner encoder, a Convolutional Encoder. The Convolutional Encoder produces two bits for every input bit.

Puncturing can then be used to throw away some of the generated redundant bits. The overall code rate is selectable by jumpers to be $\frac{1}{2}$, $\frac{2}{3}$, $\frac{3}{4}$, $\frac{5}{6}$ or $\frac{7}{8}$, to accommodate different data rate and error resilience needs.

The QPSK Signal Mapper then generates the QPSK signal from the Puncturing output and feeds it into two 4 times oversampling Raised-Cosine filters.

The whole Baseband Processor was implemented in a Xilinx Spartan2 XC2S200 device. We used the B3-SPARTAN2+ FPGA Board from Burch Electronic Design [1]. The device is about 10% full and achieves more than 80 MHz clock rate. This results in a symbol rate of up to 20 MSymbols/s. Figure 3 depicts the FPGA Board together with the D/A Converter Daughter Board.

We plan to increase the maximum symbol rate in the future. The current design has two speed bottlenecks. The oversampling filters may be sped up by changing the architecture. The Reed-Solomon encoder takes 16 clock cycles to encode a single byte, because it uses only one Galois Field Multiply Accumulate (GFMAC) unit. It can trivially be accelerated by using multiple GFMAC units.

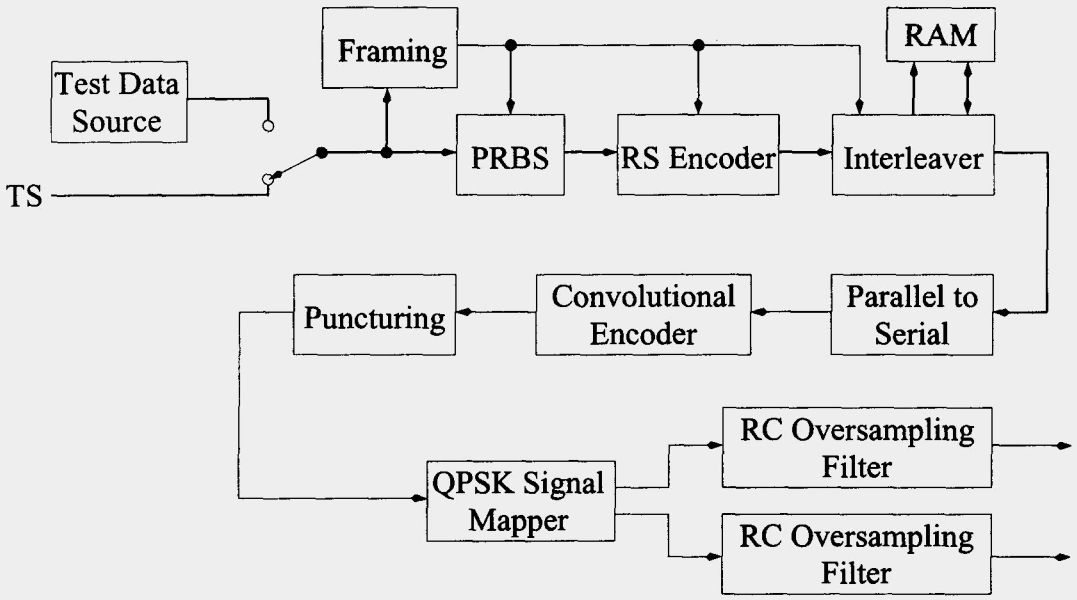


Figure 4: The DVB-S Baseband Processor

2.3 The Up Converter and PA

The QPSK RF transmitter can be realized either as a superheterodyne or as a direct transmitter. The superheterodyne concept usually achieves better carrier and sideband suppression because the operating frequency of the modulator is lower (typically below 100 MHz). But for QPSK the performance of modern integrated IQ modulators is sufficient even at 2.4 GHz; therefore a direct transmitter has been chosen which does not need bandpass filters and has thus less components and no tuning elements.

To simplify the modulator design we use ICs with integrated 90° phase shifter. Examples are RF2422 (RFMD, 0.8-2.5 GHz), AD8346 (Analog Devices, 0.8-2.5 GHz), PMB2201 (Infineon, 0.8-1.5 GHz) or MAX2721 (Maxim, 2.1-2.5 GHz). The RF2422 has been selected due to its wide bandwidth and easy availability. We also tested the MAX2721 for 2.4 GHz but had severe problems with instability[1]. The RF2422 worked very well, with some 30...35 dB carrier and side band suppression, but it is sensitive to negative spikes on the modulation inputs. After replacing it twice (and repairing some of the traces) we found the problem and added the clamping diodes in the circuit diagram. The AD8346 is considered for future trials.

The oscillator uses a MAX2620 VCO with some SMD components for the tank circuit. It is tunable between 1200 and 1300 MHz with 0-5 V tuning voltage. A slightly stronger coupling of the varicap may be useful for some extra tuning range. The LM2331 PLL (National) has been selected because it was available; in fact, most 2 GHz PLL circuits will do the job. A 2.4 GHz version of the transmitter is under development; it uses a MAX2753 fully integrated VCO and a LM2330 PLL. A PIC16F84 programs the PLL in 250 kHz steps. The loop bandwidth is about 1 kHz.

Sufficient isolation between modulator and VCO is crucial for low VCO pushing and thus low modulation distortion. Although the MAX2620 has a built-in buffer with about 35 dB isolation we added an attenuator and an extra INA340 (Agilent). For the same reason a shielding plate covers the VCO circuit and screens it from radiation from the output amplifier. The whole circuit is placed in a tin-plate box with feedthrough capacitor for the supply voltage and lowpass filters (consisting of ferrite beads and small capacitors on the backside of the PCB) on the modulation inputs. In this configuration the box can be operated close to the transmit antenna. Figure 5 depicts the IQ modulator and VCO module.

The output amplifiers uses wide band MMICs with strong internal feedback – not because they are so easy to use [2], but because the feedback improves their linearity. The output stage NGA-489 (Stanford Microdevices) has an extraordinary high IP3 = +39 dBm with only +17.5 dBm 1-dB-compression and 80 mA @ 5 V supply. At +13 dBm (20 mW) average output power or about +18 dBm PEP an ACPR (adjacent channel power rejection) of 50 dB is achievable.

The circuit is realized on a ordinary two layer FR4 board with a ground plane at the back side.

The PA should just amplify the signal without adding too much distortion. It depends on the ACPR requirements how difficult this task will be. If 40 dB or more is necessary – imagine an ATV repeater with omnidirectional antenna in the vicinity of other services in the same band – either a well-designed class AB amplifier with optimized linear 24 V transistors has to be used, or a class A amplifier. For our demonstrator we modified a M57762 (Mitsubishi) 'brick' by removing the internal bias diodes and adding external active bias circuits to each of the three stages. Operated at 4 A @ 13 V bias with a large heat sink it delivers 3 W (average) output power with only 10 mW drive and 40 dB ACPR. Figure 6 shows the output spectrum of the PA.

Instead, a class AB linear amplifier of 'SSB quality' (about 20...25 dB IM rejection) can be used followed by a bandpass filter to clean-up the spectrum. This filter can be realized in aluminium as a comb line or interdigital line resonator structure for a bandwidth of abt. 10 MHz if 1...2 dB insertion loss and

[1] The Maxim EV kit uses a 4 layer board and blocking capacitors of 0402 size

[2] In fact, the amplifier chain was oscillating at 10 GHz during the first tune-up of the circuit

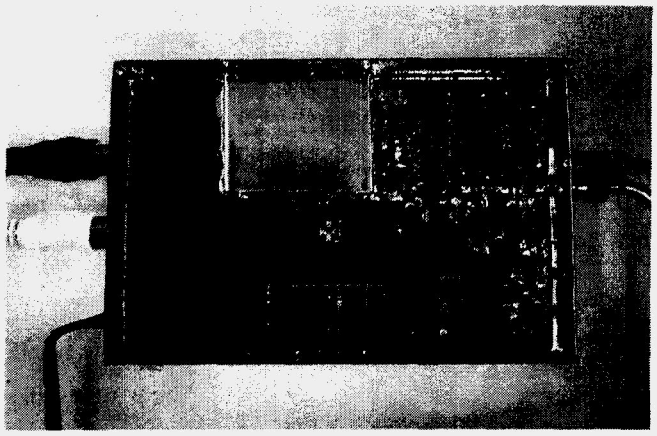


Figure 5: The IQ-Modulator and VCO Module

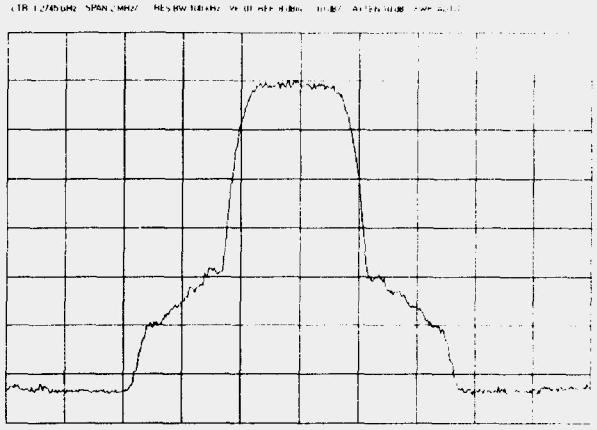


Figure 6: The Power Amplifier Output Spectrum

a careful tuning for minimum group delay distortion are accepted. For smaller bandwidths (low symbol rates), designing such a filter with low insertion loss and good thermal stability is more challenging.

For higher microwave bands, e. g. at 10 GHz, these spectral issues are usually less important for radio amateurs so that amplifiers of only moderate linearity can be used too. A transmitter can be built by up-converting a 1 or 2 GHz modulated signal; but a 10 GHz direct IQ modulator with a tuned 30 dB carrier suppression has also been published already [9].

3 Conclusion & Outlook

In this article, we presented a Digital Amateur TeleVision (D-ATV) system. The system consists of a commercially available set top box and a custom transmit chain. Using state of the art components, we were able to complete the transmit chain as a spare time project in only two months.

Figure 7 depicts DVB-S transmitter system we demonstrated at the Friedrichshafen Ham Fair in June 2001 [2]. Table 3 lists the parameters used for the demonstration. Several cheap satellite TV receivers distributed throughout the fair hall received the signal.

Given sufficient interest, Stefan Reimann, DG8FAC plans to produce a small quantity of the transmit chain boards.

Since the Xilinx FPGA device is now only about 10% full, there are numerous extensions possible, such as

- multiplexing several sources onto the same carrier
- increasing the maximum symbol rate of the Baseband Processor
- filling unused MPEG2 transport frames with network data, such as AX.25 or TCP/IP traffic

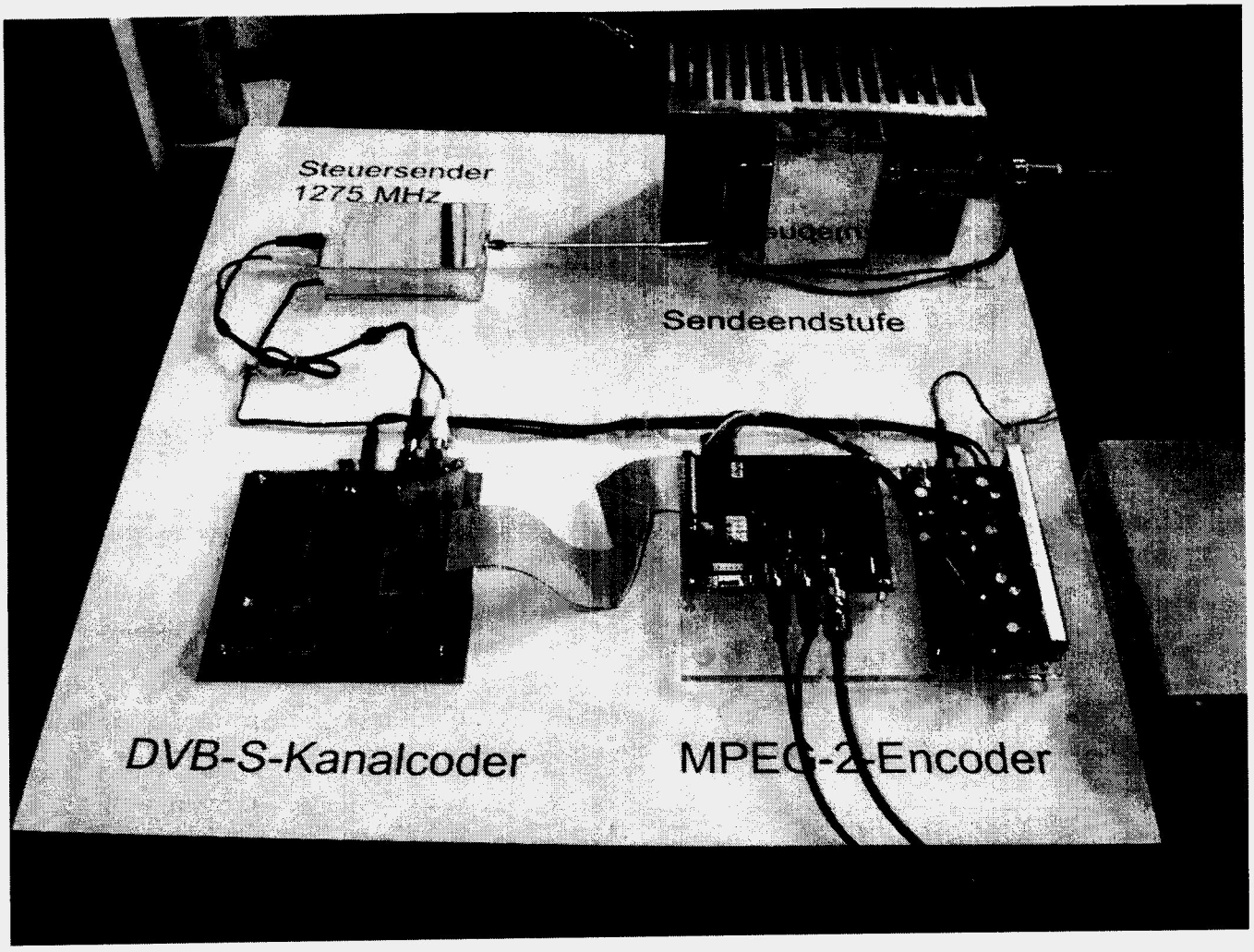


Figure 7: The Complete DVB-S Transmitter demonstrated at the Friedrichshafen Ham Fair

Transmit Frequency	1275 MHz
Symbol Rate	3 MSymbols/s
Bandwidth	4 MHz
Convolutional Code Rate	$\frac{5}{6}$
MPEG2	MP@ML D1 4:3
	25 Frames/s
	4.5 MBit/s
Transmit Power	2 Watts
Antenna	vertically polarized, omnidirectional

Table 1: Friedrichshafen Demonstration Parameters

References

[1] Burch Electronic Design. http://www.BurchED.com.au.

[2] Digitales Amateurfernsehen nach DVB-Standards. http://www.d-atv.de/.

[3] European Telecommunications Standards Institute (ETSI). http://www.etsi.org.

[4] SR Systems. http://www.sr-systems.de.

[5] ISO/IEC 13818-1 Generic Coding of Moving Pictures and Associated Audio: Systems Recommendation H.222.0, 04 1995.

[6] ETSI EN 300 421 V1.1.2 Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz satellite services (DVB-S), 08 1997.

[7] ETSI EN 300 429 V1.2.1 Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for cable systems (DVB-C), 04 1998.

[8] ETSI EN 300 744 V1.4.1 Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television (DVB-T), 01 2001.

[9] Matjaz Vidmar. *DUBUS Technik-Buch V*, chapter No-Tune SSB Transceivers for 1.3, 2.3 , 5.7 and 10 GHz, pages 203–291. DUBUS, 1995–1997.

APRS in Hollywood
Integrating Real Time 3D Graphics with Wireless GPS systems.

Phil Brock
Senior Graphics Programmer
REZN8, Hollywood, CA, USA.
Pbrock@silcom.com

Bill Kovacs
Chief Technology Officer
REZN8, Hollywood, CA, USA.
Bkovacs@rezn8.com

Darryl Smith, VK2TDS
Radioactive Networks,
Sydney, Australia
Darryl@radio-active.net.au

Abstract

This paper describes the integration of a real time wireless GPS/GIS system with high end real time 3D computer graphics. It describes some of the infrastructure required for such a system, and the work required for implementation.

Introduction

A large percentage of the worlds information is geographically based. Applications such as the graphical APRS clients are a great way to view this data, but the information although dynamic lacks impact. With this in mind, REZN8[1] and Radioactive Networks[2] joined forces to create a demonstration on how geographical based data such as that from APRS could be rendered if combined with the fastest desktop computing power, the fastest graphics cards and the latest 3D rendering packages.

We are now to the point where realistic 3D graphics can be generated in real time on the desktop – and not just the graphics like those found in Doom or Quake but broadcast or film quality. This project was done using APRS as the base which all other technologies were added. The technology we are presenting here is quite intensive on animators and modelers at the moment, but developing technology should reduce the resources required.

The Idea

In the outside world the way that information is presented to the user is paramount. If it were not, Linux would be the predominant desktop operating system and Windows would be a cute toy for hackers. Like most software of it's type, all the APRS software under windows environments presents the information in 2D.

REZN8 wanted an application they could use to highlight their graphics expertise when integrated with the latest microprocessors and nVidia graphics cards. They decided that adding 3D graphics to APRS would be the ideal example of this expertise and technology.

Previously, REZN8 had produced a 3D model of Sydney for the Paralympic Marathon. This model had a line moving through the city growing as the athletes moved. This line extended at a constant speed and had to be adjusted by hand as the race progressed. A photo of these graphics can be found in figure 1.

For this presentation we decided that extending this concept to the next level – Adding live wireless GPS data for dynamic generation of graphics.

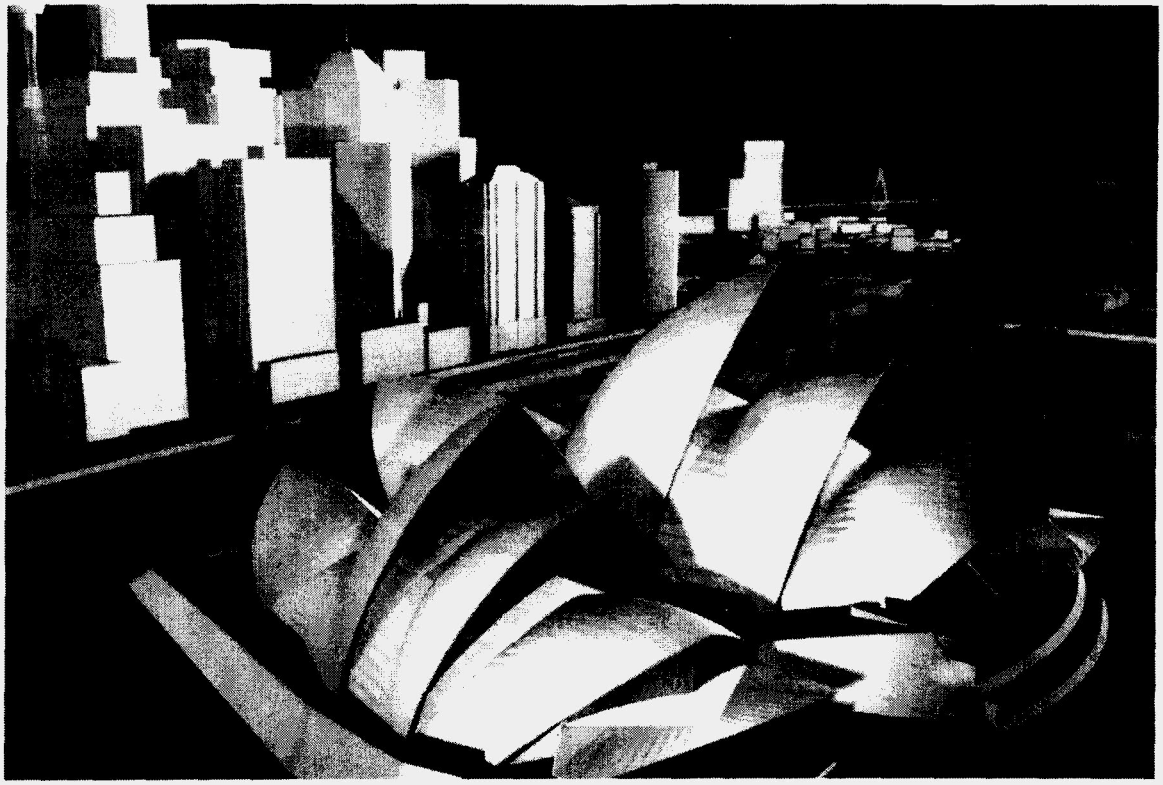


Figure 1 - 3D Animation of Sydney Paralympic Marathon

The Concept

To give the demonstration some appeal, we needed a gimmick – and the obvious gimmick was with the object that we were tracking. After looking at a few options we settled on children's scooters that have become popular recently.

We decided to have two people race the scooters outside the venue of our demonstration, whilst being tracked on a video screen inside. To make the race a little more exciting, one of the contestants took a somewhat shorter alternate route.

During most of the race the only image shown was a live 3D representation. At the end, closed circuit television was used to show that this was not faked.

Tracking the scooters

GPS was used to track the scooters. This was not ideal for low speed vehicles in close proximity but it performed well enough.

In order to operate flawlessly we had already chosen radio transmitters with a 50W (max) power output so that we could gain range and cut through any interference experienced.

This was important as the original concept called for tracking cars over a larger area where the power would be needed for the range.

The scooters we ultimately decided to use for the demonstration were the larger battery powered units. But with these units we had the problem of where to mount the radio, GPS and controller. We found that if we removed the internal battery, making the scooters self powered, we could mount most of the equipment in the battery compartment.

Equipment

We decided to use Kenwood D-700 transmitters with integrated TNC's as they were tightly integrated making them less likely to fail. The radio was connected to an AISIN GPS receiver purchased surplus as well as to a custom controller for triggering transmissions. These three objects were mounted in the now empty battery compartment.

With this much equipment in the battery compartment there was no room to mount a battery with the equipment. We eventually mounted a 7 AH battery on the back of the scooter with Duct Tape. [If we had been allowed to make major modifications we could have fit the battery in, but our sponsor wanted to give the scooters to his children after the demonstration]

The D700 front panel was mounted on the handle bars, along with the GPS antenna. A small magnetic mount was also mounted on the handle bar column.

With the D-700 in POSITION mode, this made the scooters the ultimate toy with the GPS based speedo function. Several people we showed the scooters to believed that we should just sell these and make a killing – even more so when they realized that they would know where their children were at all times.

Technical Implementation

In order to track the scooters in near real time, we needed to have the D700 transmit at least every 2 to 3 seconds. Every second would have been better, but every 2 to 3 seconds was acceptable. Unfortunately the quickest beacon rate of the D700 is 10 seconds, which is enough for most applications, but not ours.

The D700 radio has a GPS port for parsing GPS data in many formats. In TNC mode, there is a command called LTMON which allows the GPS data to be monitored automatically on the serial port. For instance when the command LTMON 3 is issued, GPS data will be sent to the serial port every three seconds.

The GPS data is returned in a string starting $PNTS, a proprietary format designed by the makers of the chipset in the Kenwood radio. Since our application was for customized receiving software, any line starting $PNTS was simply sent over radio using converse mode.

We implemented this using Rabbit Semiconductor microcontrollers attached to their development boards. In a production environment we would probably choose a different processor.

Data Reception

The receiver was a Kenwood TM-251 radio connected to a Kantronics KPC9612 TNC. Since we were only receiving at 1200 bps, both these pieces of equipment could have been downgraded but allowed for reconfiguration if required.

The TNC was connected to a 150 foot CAT-5 serial cable onto a stage where the demonstration of this technology was taking place.

In the field, a D7 handheld with internal TNC was used to monitor the performance of the scooters so that we could fix any problems. After operating perfectly for a few days before hand, we actually needed to reprogram one of the TNC's just before the demonstration.

Server Software

The server was responsible for reading GPS data from the rs232 connection, processing the input stream and separating the two streams of data. The software also moved the GPS coordinates into the coordinate system used by the 3D model – by stretching, rotating and scaling the coordinates. It also acted as a TCP/IP server for clients to connect to.

We extended this a bit during testing when we found that GPS data was not a perfect match for a small vehicles on a short race – so we placed a model of the path into the server as well. This allowed us to use least squares approximation to find the closest point on the path to the GPS data if required.

It also allowed us to build extrapolation for the GPS data to remove the jerks in movement – which is important when video moves as 30-60 frames/second, and the GPS data was arriving at 3 second intervals.

Having a server allowed us to run more that one client displaying the graphics. This was important since we were beta testing hardware for the highest performance. It also allowed us to tune the incoming data to the format required by the graphics engine client software. We could also play back GPS data to the clients in case of system failure.

Client Software

For each valid data point, the server processes the data and sends to the client.

The client, programmed in C++, allowed for dynamic control of the viewing location with the keyboard and mouse. The controls included pan, tilt, zoom etc. just as if this were a real camera. The client also stored a copy of the 3D database as well as the Maya 3D real time graphics engine.

This software required a copy of the 3D model on the local PC to improve processing. The software that we wrote has the potential to revolutionize the GIS industry.

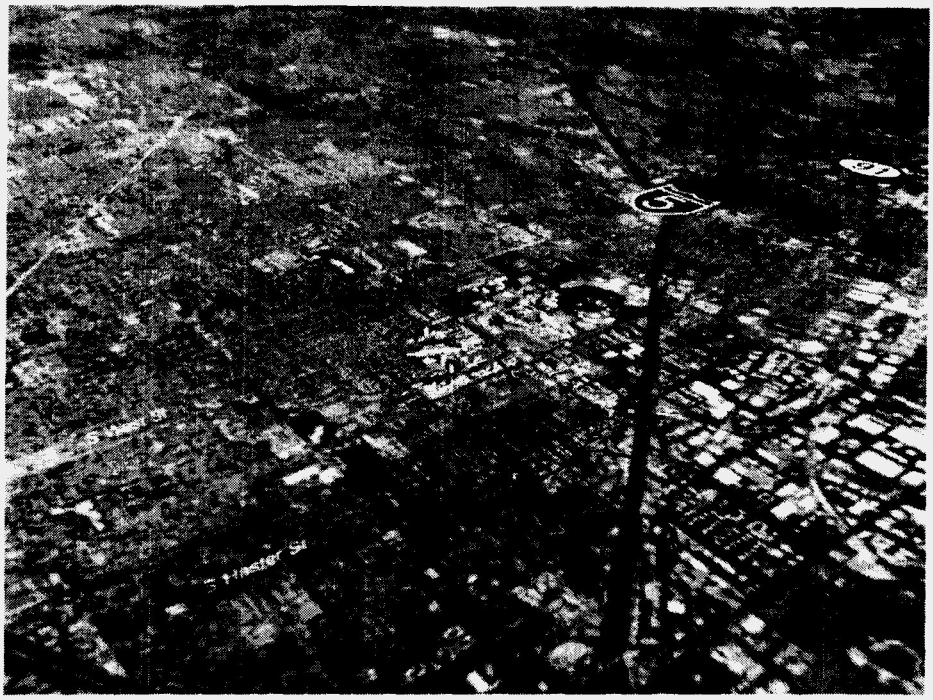


Figure 2 - LA before zooming into the venue

Computer Modeling

Whilst it is outside the scope of radio, creating a 3D model of the area is probably the most important task of this exercise. I am detailing it here to give an idea of the sheer amount of work involved at the moment.

The first step was to commission some aerial photography, of the whole area of interest in general, and closer up images of the area of the demonstration. These photographs were provided as large (200+ Mbyte each) TIFF files. Using the 3D Studio Max and Maya we were able to create a seamless transition so that when we zoomed into the area of interest, the higher resolution image was inset. Figure two shows these files.

Although this only created a 2D representation of the area, looking onto the map side on did look 3D until zoomed in close. This is when the magic really happens.

With the 2D representation as a backdrop, the REZN8 artists created a 3D model of the local area by hand. This involved adding everything from roads to buildings and letter boxes.

This work was assisted with lots of photographs taken from ground level as well as some artistic license. This sort of effort is quite labor intensive, and not the sort of task to be undertaken lightly.

Discoveries and Challenges

When we started analyzing the problem of viewing the data we came across the height information from the scooter – or more correctly found that the GPS sentence that we were using did not contain height data. Upon analyzing the problem, we worked out that the height data was not useful to us anyway.

The height data from GPS contains a higher error than the latitude and longitude. More importantly, in most cases when visually modeled, the height data needs to be the most accurate. It is not easy to tell if an object is one meter too far to the left or the right, but it is very easy to tell if they it is one meter off the ground, or embedded one meter into the ground.

When we thought about the problem, there was one obvious solution – we should always have the moving object at exactly ground level. Since scooters do not fly despite what you may have seen in 'Back to the Future', this was perfect.

The GPS receivers that we were using performed extremely well in most cases. However they do not like to change hemispheres of the world without a hard reset. We spent well over an hour on top of a Hollywood car park before discovering this.

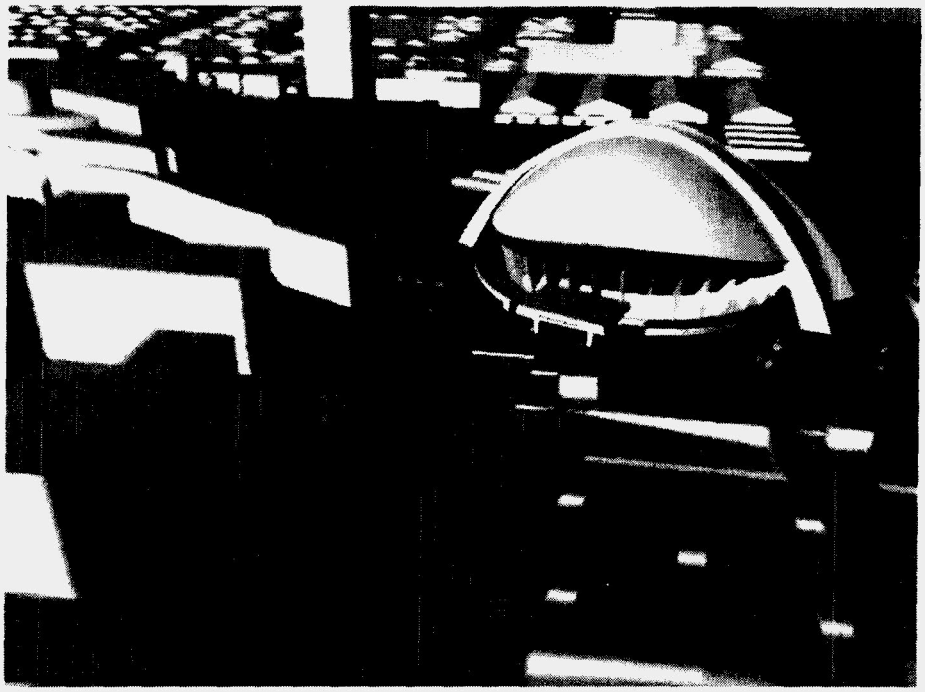


Figure 3 - Local area with single object being tracked near the entrance to the dome building.

Conclusion

Our demonstration of this technology was successful, with all those who saw it leaving very impressed. We have once again shown that the resources needed to produce the type of graphics described are significant, even on such a small scale.

Having said this, we believe that it will soon be possible to integrate real time 3D graphics with real time GIS data from systems such as APRS without having to dedicate so many resources.

We have shown what is possible. The challenge now is to improve it and decrease the cost.

[1] REZN8 is a Hollywood based studio specializing in visual effects for film and television, commercial production, internet design and programming, interactive presentations, as well as hardware and software integration. For more information visit http://www.rezn8.com

[2] Radioactive Networks is an engineering consulting based in Sydney Australia. Specializing in wireless networking solutions.

Deconvolution in Communication Systems

The title contains a fancy name for a DSP process that can reveal details of wave propagation paths without prior knowledge of their nature. Multipath and other distortions may be detected and corrected to some extent with this technique. It's also useful in DSP filter design. Come discover how it's done.

By Doug Smith, KF6DX

Multipath distortion is the enemy of many radio communicators, whether they are interested in moon-bounce, terrestrial microwave or HF. Multipath may be generally described as a situation in which radio signals take many different routes between transmitter and receiver. Those routes quite often have different lengths, so received information consists of a multitude of superposed copies of the transmitted information, smeared over time.

It might seem at first that there is nothing DSP or any other technology can do about that distortion, since it is caused by physical phenomena beyond our control. But where knowledge exists beforehand about the nature of the transmitted signal, it turns out something often can be done. I'll try to explain what I've learned about that.

225 Main St
Newington, CT 06111
kf6dx@arrl.org

Modeling Multipath Environments

Imagine you're standing just inside the Taj Mahal (Fig 1). The clack of shoes meeting tile and the hush of whispers bounce lightly from the walls and ceiling. Your friends entered just moments ago. You scan the foyer; they aren't in sight. You call out to them. Your voice echoes through the halls and chambers of that place for what seems like an eternity—until security personnel come and tell you not to do that!

Your friends, having reached a distant part of the building by now, hear the sound; but it doesn't sound much like you. In fact, it sounds more like a dull roar because your voice has taken so many paths to their location. All the echoes overlap so much that words and even syllables are indistinguishable. You are in a *reverberant environment*.

Your friends begin walking toward you. As they come closer, you speak again. This time, they understand you and reply. The number of paths and the differences in their lengths have now decreased; the time smearing and overlap of echoes are now little enough to allow you to be intelligible. You have demonstrated a useful model for reverberant environments: many discrete paths, each with its own transit time or delay and each with a particular attenuation. See Fig 2.

In the figure, multipliers h_n have

values less than unity and represent the attenuation on paths whose delays are proportional to n. Notice that no signal propagates directly from the input to the output; the output is derived only from delayed signals. That indicates the usual situation: Your friends are some finite distance from you, even when in view. So even on a direct path, there is always a positive, non-zero propagation delay. The model also applies when your friends are around the corner; they cannot hear your voice directly, but only the sound that is bouncing off the walls, floor and ceiling.

When the delays z^{-n} in the model are spaced apart by the same amount of time, which we shall call the *sampling period*, the set of attenuation constants h_n is referred to as the *impulse response* of the system. In fact, Fig 2 is exactly the same as the block diagram of a *finite-impulse-response (FIR) filter*, a common construct in DSP.[1] While it is perhaps strange to think of the Taj Mahal as a filter, that is indeed what it is. When the constants h_n are chosen strategically, the system may be made into almost any filter shape imaginable. When they are undefined, as in the case of sounds propagating through buildings or radio signals through whatever medium, the transfer function (frequency response) is also undefined.

If the impulse response of a system can be found (the model), then another system may be built having a transfer function that is the inverse (the inverse model) of the original system. When the two systems are cascaded, the final output is a restored version of the input signal (the desired). For the Taj Mahal or a set of radio propagation paths, the hard part is discovering the original impulse response. When the environment is known and fixed (as in the Taj Mahal), the impulse response may be discovered by modeling the structure and doing ray-tracing experiments on a computer, for example. When the environment is unknown (a radio path), we must resort to *inverse modeling* to get a clue about the corrupting system's impulse response.

That is fairly easy when the unknown environment is fixed. When it is changing, it is much more difficult. Even then, though, DSP provides weapons to combat the enemy. Follow me into a discussion of how those two cases are generally handled.

Inverse Modeling

When performing the operation shown in Fig 2, the output is the sum of all the delayed, attenuated signals. That sum is called a *convolution sum*; the input signal is said to be *convolved* with the impulse response. A convenient notation for the convolution sum is:

$$r_t = \sum_{n=0}^{L-1} h_n x_{t-n} \qquad \text{(Eq 1)}$$

where r_t is the output at discrete time t, x_{t-n} is the original input signal at time $t-n$, and L is the length of the finite-impulse response. The transfer function of that system may generally be found by taking the discrete Fourier transform (DFT) of the impulse response, h_n:

$$H_\omega = \sum_{n=0}^{L-1} h_n e^{-j\omega n} \qquad \text{(Eq 2)}$$

Where ω is the angular frequency in radians/s. The goal of inverse modeling is to discover the system that has a transfer function equal to the reciprocal of H_ω. Were a copy of the original input signal, x_t, available, that would be easy to do, as shown in Fig 3. The corrupted signal r_t forms the input to the inverse filter, whose coefficients are adjusted in some way based on a comparison between the original input signal, x_t and the doubly processed output of the inverse filter, y_t. When the error signal e_t goes to zero, the inverse filter's frequency response G_ω is the reciprocal of the corrupting system's:

$$G_\omega = \left(H_\omega\right)^{-1} \qquad \text{(Eq 3)}$$

Inverse filter G is said to *"deconvolve"* the original input signal and the

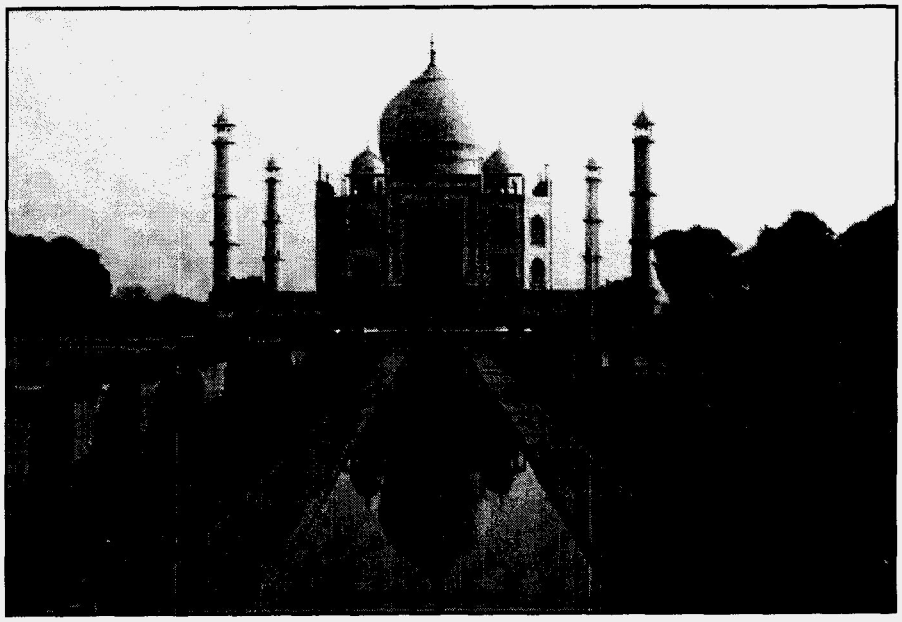


Fig 1—A reverberant environment.

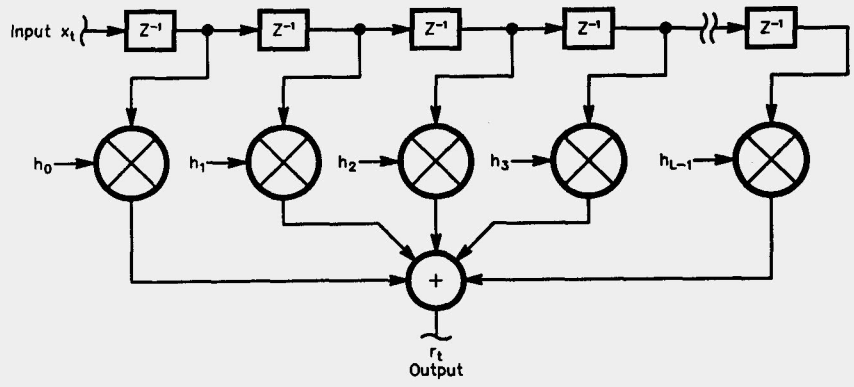


Fig 2—An FIR model of a reverberant environment.

[1]Notes appear on page 51.

corrupting filter's response, restoring the original signal. In the steady state, the inverse filter performs the operation:

$$y_t = \sum_{n=0}^{L-1} g_n r_{t-n} = x_t \qquad \text{(Eq 4)}$$

The transfer function of the cascaded system is a single, unity-amplitude impulse. A fixed delay is used in the upper path of the original input signal to compensate the delay through the two filters.

It's perhaps surprising that impulse response g_n is not necessarily the inverse DFT of G_ω, although that has sometimes been stated, incorrectly. The proof of that is a bit more complex than what I want for this article. Oppenheim and Schafer take up the subject briefly.[2]

Now it's time to mention how impulse response g_n is adjusted during inverse modeling to efficiently achieve the desired result. The most popular method is called the least-mean-squares or LMS algorithm. It was published by Widrow and Hoff in 1960[3] and it's the same as algorithms currently used for adaptive noise reduction and automatic notches in radio receivers.

In the LMS algorithm, each value or *coefficient* of the impulse response is adjusted at each sample time according to:

$$g_n' = g_n + 2\mu e_t x_t \qquad \text{(Eq 5)}$$

for L values of n, where μ is a constant chosen to alter the speed of convergence and the amount of error in the steady-state solution. Additional details of the behavior of adaptive filtering systems may be found in the Amateur Radio literature[4] and will not be treated further here.

You may be questioning how the methods described above can be useful, since they require a copy of the signal originally sent. One application is found in the suppression of echoes on telephone circuits. Another is found in DSP filter design.

Telephone-Line Echo Suppression

On a two-wire, full-duplex telephone circuit, hybrids are used at each end to segregate transmitted and received signals. The hybrids must achieve significant isolation between the two signals lest a signal transmitted at one end arrive at the other end to be retransmitted toward the sender. The result is a series of echoes. When termination impedances are not perfect on the line or imbalance exists, those echoes are always present. They are most discomfiting to the talker—and perhaps also to the listener—especially over lengthy, overseas paths having transit times of 300 ms or more. This sort of thing can also be a problem in speakerphones.

The system of Fig 3 may be employed to eliminate the echoes, since copies of both transmitted and received signals are present at the transmitter. That is, in fact, what telephone companies currently do to handle the situation. I notice some long-distance companies need to check the operation of their echo cancelers. Echoes were rampant in the early days of long distance, then the problem seemed to have virtually disappeared for a long time; but now, I regularly get reports of its reappearing.

LMS Filter Design

This example is one of direct modeling rather than inverse modeling, so it's a little different from what we've covered so far; but it's still useful because it shows something about the underlying concepts of system modeling in general.

Imagine we want to find the finite impulse response corresponding to some particular filter shape—say, a low-pass. First, we must characterize the desired transfer function, H_ω, completely by both its amplitude and phase responses. Amplitude versus frequency is more important at this stage than phase; we dream up a *pseudo-filter* having the desired response. FIR filters generally have linear phase responses; the phase versus frequency plot is a straight line. We place this pseudo-filter into the modeling system shown in Fig 4. Notice that the pseudo-filter need not actually exist as an FIR filter; it is just a block that replicates the transfer function we want, and we may perform that function in any way.

To make the adaptive filter, G, in Fig 4 converge to match the pseudo-filter's response, the generator signal x_t's spectrum must contain energy at all frequencies of interest. White noise is a good first choice for this signal source. Start the thing going and when the LMS algorithm has minimized the error e_t, the adaptive filter will have converged on the impulse response most closely matching our desired response.

Depending on the length (L) of the adaptive filter, it may be difficult to achieve the desired response at certain frequencies exactly. That may be addressed in LMS filter design by changing the amplitude-versus-frequency content of the generator, x_t. A large relative amplitude of the generator's content at some particular frequency allows the filter to more closely meet its specification at that frequency.

When a Desired Response Is Not Available

For radio signals, the telephone scenario above is not particularly relevant. The question becomes "How can we use inverse modeling when a copy of the original signal is not available?" The answer is that the signal x_t used to compute error signal e_t and used in the LMS algorithm need not be an exact copy of the original; it need only be a reasonable approximation of that signal. Any information about the original is useful in nudging the algorithm toward convergence at the start of adaptation; we then get a better deconvolution that, in turn, helps the next iteration toward the optimal solution. Let's look at some examples that illustrate how to make inverse modeling work without an exact copy of the original signal.

Adaptive Equalization of a Dispersive Medium

A *dispersive* medium is one in which different frequencies travel at different velocities. That is, the *group delay* is not constant. To grasp these terms, let's say we have a medium or channel with frequency response H_ω. Response H_ω may be completely characterized

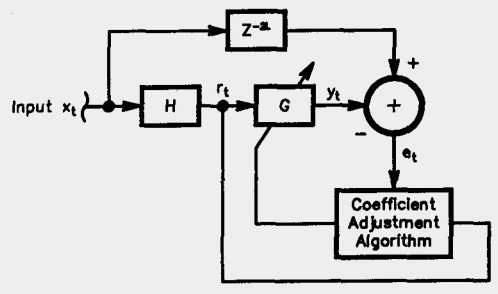


Fig 3—Block diagram of an inverse-modeling system.

by its amplitude response, A_ω, and its phase response, φ_ω:

$$H_\omega = A_\omega e^{j\phi_\omega} \quad \text{(Eq 6)}$$

The time delay (or phase delay) through the channel is:

$$t_{prop} = \frac{-\phi_\omega}{\omega} \quad \text{(Eq 7)}$$

and the group delay is equal to the differential time delay:

$$t_g = \frac{-d\phi_\omega}{d\omega} \quad \text{(Eq 8)}$$

A medium is said to be dispersive if the group delay is not a constant function of frequency; that is, if:

$$\frac{d^2\phi_\omega}{d\omega^2} \neq 0 \quad \text{(Eq 9)}$$

Dispersive propagation is very similar to multipath, since it also implies that received information is smeared over time. Now let's say that we have a dispersive medium that is not horribly so. We also stipulate that noise levels are reasonably low, so as not to be a problem in demodulation of the data signal we're going to send through the medium. To further simplify what follows, let's also say the channel has a very large bandwidth.

Adaptive equalization may conveniently be discussed by considering the case of a single carrier, modulated by a single binary signal. While that is not a common situation on telephone lines, the format is still used over radio quite a bit. In any case, it is the simplest instance, and study of m-ary or multiple-carrier systems stems from it.

Now a simple data transmitter encodes a binary one as a transition of one polarity; a binary zero is encoded as a transition of the opposite polarity. That is true no matter the modulation format. FSK, PSK and other traditional formats may employ rapid polarity transitions that, unless otherwise limited, may cause the signal to occupy a rather large bandwidth. Even through a channel of large bandwidth, dispersion alters the shape of the transitions received because the carrier and modulation sidebands propagate at different velocities. That ultimately limits the data rate that may be supported.

Let's look at what happens when a very sharp one-zero transition passes through our dispersive channel (see Fig 5). What started out as an instantaneous state reversal now becomes smeared in time; its shape is determined by the impulse response of the channel. The group-delay-induced distortion makes recovery of the data more difficult. We can say that the received signal is the convolution of the original signal and the impulse response of the channel. Its spectrum is the product of the spectrum of the original signal and the transfer function of the propagating medium. In other words, convolution in the time domain is equivalent to multiplication in the frequency domain.[5]

Forward Equalization

It is often desirable to equalize the channel so that it can support higher data rates. The goal of an equalizer is

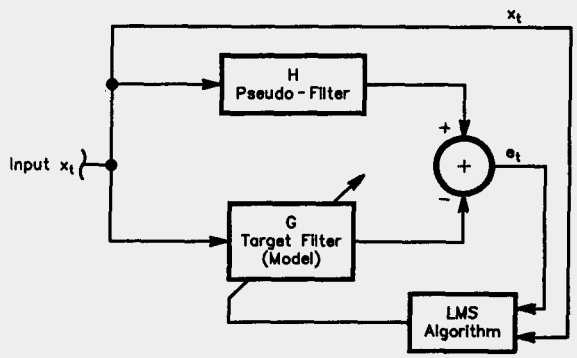


Fig 4—Block diagram of an LMS filter-design algorithm.

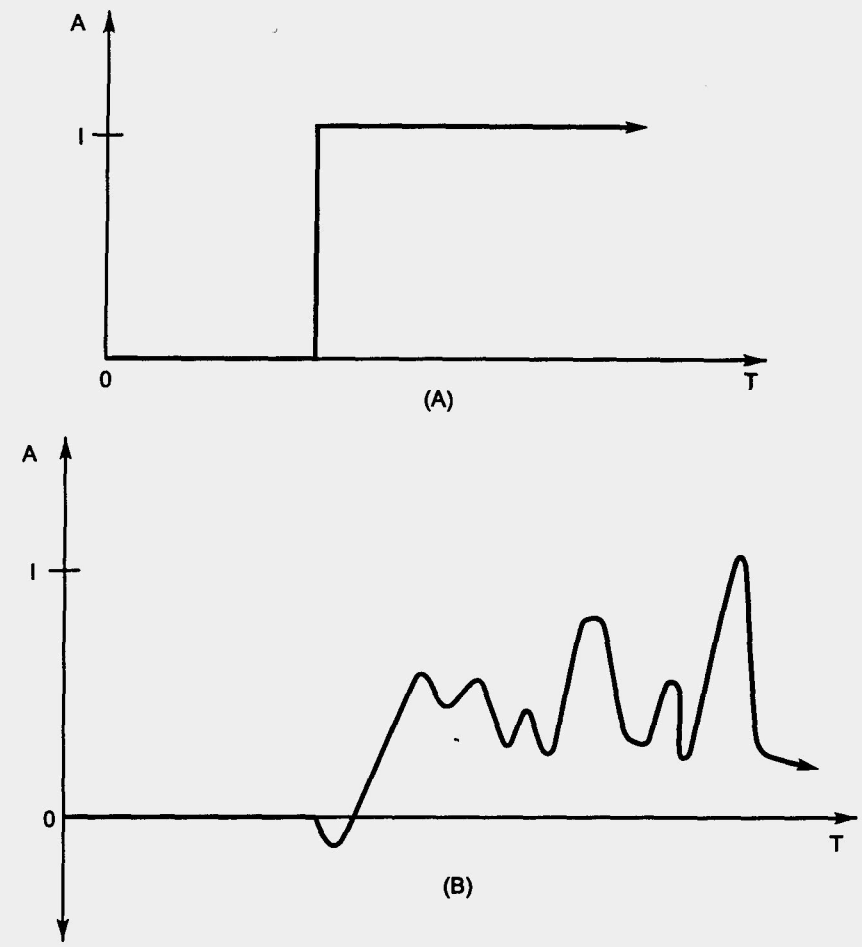


Fig 5—A: A sharp data-state transition. B: Transition as received through a dispersive medium.

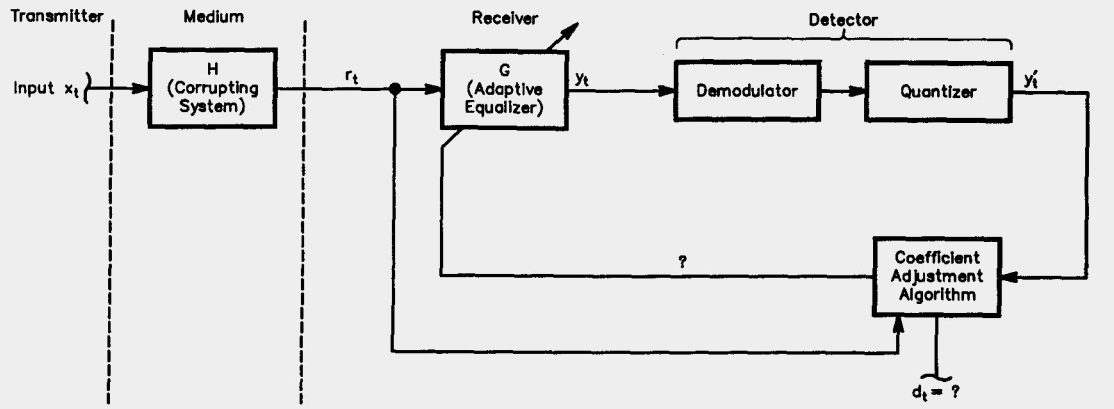


Fig 6—Block diagram of forward equalization.

to achieve a constant group delay and a flat frequency response. To equalize a channel, insert an FIR filter (G) into the data path and add a demodulator and *quantizer* at its output, as in Fig 6. So how do we decide how to adjust the equalizer? Well, one way is to arrange for a known *training sequence* to be transmitted and to compare the equalized signal with a locally generated copy of that sequence. The LMS algorithm may be used to adapt the equalizer. Then the block diagram is as shown in Fig 7. That system is fine for channels whose conditions do not change rapidly, as long as retraining can be tolerated periodically.

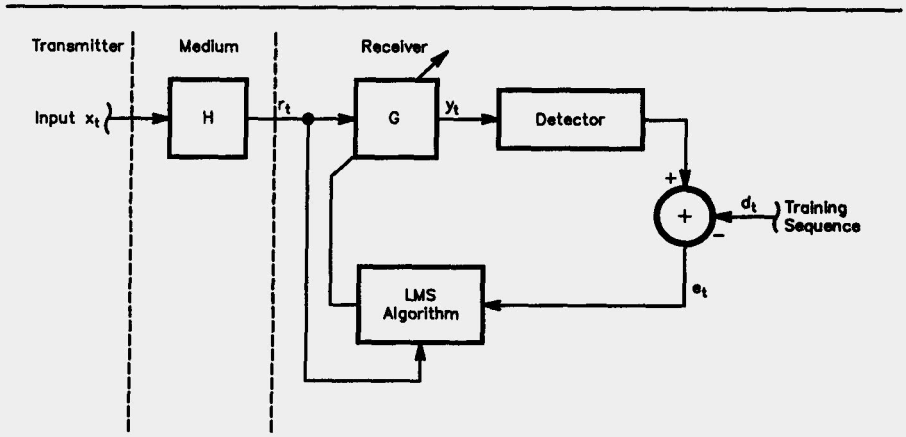


Fig 7—Forward equalization using a training sequence.

Decision Feedback Equalization

A method for deriving d_t using only the adaptive filter's output was discovered by R. W. Lucky of Bell Laboratories,[6] obviating the need for *a priori* knowledge of the original signal. Lucky (an apt name!) found that d_t could be approximated by the demodulated signal itself, as shown in Fig 8. The bet is that if the dispersion is not severe, the demodulator generates bit decisions that are close to correct, and the adaptive filter moves toward the correct solution.

As it happens, this system works well when not much noise is present and when the dispersion is mild. Performance is improved when the sampling rate is increased beyond just once per bit.

That is fine for digital signals, but what about analog signals? The decision-making process is much tougher in that case; but the processes of *linear prediction* and *autocorrelation* may be used to steer the algorithm. Details of that are beyond the scope of this paper. For more information, refer to Widrow and Stearns.[7]

Homomorphic Deconvolution

Now that is an esoteric phrase, but what does it mean? Well, it's a method of deconvolution that uses nonlinear transforms of signals, which are manipulated algebraically. More specifically, the nonlinear transform used is the logarithm. I'll show how a useful property of logarithms reduces multiplicative systems to simple superposition and why that is useful in decon-volving signals.

As mentioned before, convolution in the time domain is equivalent to multiplication in the frequency domain. When a signal passes through a propagation medium, the spectrum of the convolved signal is the product of the original signal's spectrum and the frequency response of the medium. For an input signal x_t having spectrum X_ω and a medium having impulse response h_n and frequency response H_ω, a convolved signal y_t has spectrum:

$$Y_\omega = X_\omega H_\omega \qquad \text{(Eq 10)}$$

Now for that useful property of logarithms, which is:

$$\log(ab) = \log a + \log b \qquad \text{(Eq 11)}$$

If we take the logarithm of Y_ω, we have:

$$\begin{aligned} \log(Y_\omega) &= \log(X_\omega H_\omega) \\ &= \log(X_\omega) + \log(H_\omega) \\ &= C_{x_\omega} + C_{h_\omega} \end{aligned} \qquad \text{(Eq 12)}$$

Taking the inverse Fourier transform of Eq 12 therefore results in the sum of two time-domain sequences:

$$c_{y_t} = c_{x_t} + c_{h_t} \qquad \text{(Eq 13)}$$

c_{yt} is called the *cepstrum* of y_t. That term and a bunch of other funny terms were coined in a paper by Bogert,

Healy and Tukey.[8] The block diagram of a system that produces it is shown in Fig 9.

Eq 13 is a useful result since, in many reverberant environments, the two cepstral components c_{xt} and c_{ht} are easily separated because they are so different. For example, let's say that most of the energy in c_{ht} lies at low values of t and most of the energy in c_{xt} lies at high values of t. That might be the case for a voice bouncing around in the Taj Mahal. Simple window functions may segregate the individual energy contributions. Then each cepstral component may be transformed back to a regular time sequence using a process that is the inverse of Fig 9. That inverse system is shown in Fig 10 for one of the components, c_{xt}. Its output is a deconvolved (restored) version of x_t.

Homomorphic deconvolution requires minimal information about the nature of the original signal and of the propagation medium. The basic requirement is that the significant length of the medium's impulse response be considerably different from the rates of change in the original signal. Where echoes are spaced at a constant period, the contribution of c_{ht} may be removed with a window that looks like a notch filter, removing only those samples that fall within a small range of values of t. In that last case, though, a less-complex method may exist for *de-reverberation*.

A Sigma-Delta Method for De-Reverberation

In the special case where all echoes are spaced apart in time by a constant amount and those echoes decay in amplitude geometrically, a more straightforward method may be used to recover the original signal. Such reverberant environments may be found in radio communications systems and in public-address venues like large baseball or football stadia, for example. You may hear the announcer get on the public-address system and say, "Now batting, batting, batting...number nineteen, nineteen, nineteen...Tony Gwynn, Gwynn, Gwynn!"

The sound you hear is the sum of the direct signal and an infinite series of regularly spaced echoes declining in amplitude exponentially. The situation may compactly be represented as a summation:

$$y_t = \sum_{k=0}^{\infty} \mu^k (x_{t-nk}) \qquad \text{(Eq 14)}$$

where x_t is the input signal, μ is a positive constant less than unity and n is the number of sample times between echoes. This is clearly a causal system since output depends only on present and past samples. The original signal may be recovered using "first-differencing" (discrete differentiation):

$$y_t - \mu y_{t-n} = \sum_{k=0}^{\infty} \mu^k (x_{t-nk}) - \mu \sum_{k=0}^{\infty} \mu^k (x_{t-nk-n}) = x_t \qquad \text{(Eq 15)}$$

This is approximately the difference between samples of the corrupted signal spaced one echo-time apart; but this method ignores all echoes but the first. We obviously have to wait for that first echo to occur to retrieve its energy. The system introduces a delay of n sample times before producing the desired output. Additional energy contained in all subsequent echoes is lost with this algorithm.

The total signal amplitude contributed by any particular original sample is the sum of the direct signal and all its echoes, which, assuming the original signal is of unity amplitude, is:

$$E_{\text{total}} = \sum_{k=0}^{\infty} \mu^k = 1 - \frac{\mu}{\ln \mu} \qquad \text{(Eq 16)}$$

That means that when μ=0.93, less than one tenth of the total energy is recovered by using only the first echo —a lot of the energy is in the other echoes. Signal-to-noise ratio (S/N) would be degraded by about 20 dB. In this case, it's clearly worth an additional wait to improve our lot.

To recoup the energy in all echoes would take an infinitely long period, so

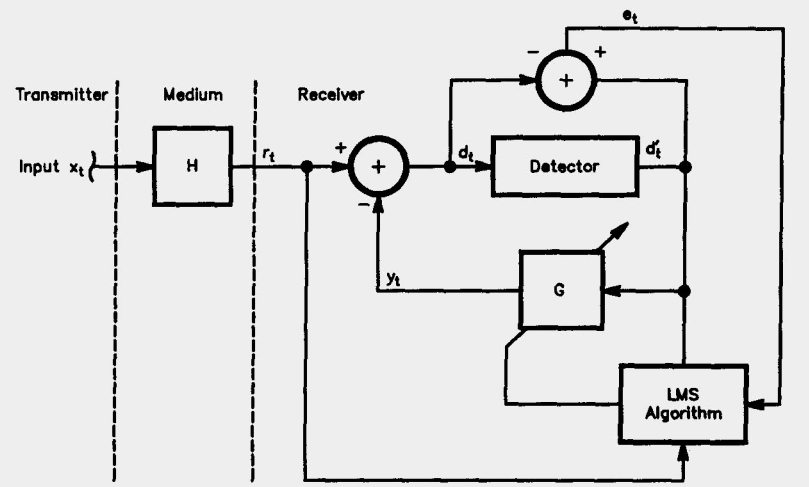


Fig 8—Decision feedback equalization using the received data as a desired signal. In this method, an adaptively filtered copy of the detected signal is subtracted from the unmodified received signal to cancel intersymbol interference. D′ is the output. Feedback equalization is typically used in concert with feed-forward equalization. For more detail, see Reference 7.

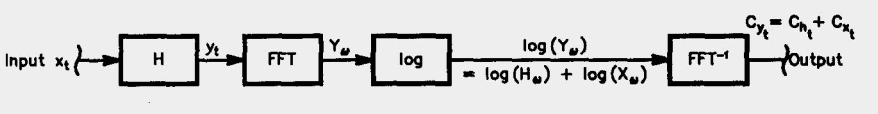


Fig 9—Block diagram of a cepstral transform.

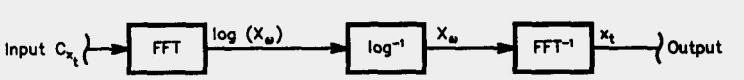


Fig 10—Block diagram of an inverse cepstral transform.

we can only regain the energy over some number of echoes, L, for which we are willing to wait. The delay incurred is nL sampling periods. To recover the energy contained in echoes beyond the first, consider taking "second-differencing," or the weighted difference between sam-ples two echo times apart. We have:

$$\begin{aligned} y_t - \mu^2 y_{t-2n} &= \sum_{k=0}^{\infty} \mu^k (x_{t-nk}) - \mu^2 \sum_{k=0}^{\infty} \mu^k (x_{t-nk-2n}) \\ &= x_t + \mu x_{t-n} \\ \therefore \mu x_{t-n} &= y_t - \mu^2 y_{t-2n} - x_t \end{aligned} \quad \text{(Eq 17)}$$

where x_t is determined by Eq 15 above. x_{t-n} was x_t n samples ago, and is added to the result of Eq 17 to yield $x_{t-n} + \mu x_{t-n}$. A similar operation may be performed for $y_t - \mu^3 y_{t-3n}$, $y_t - \mu^4 y_{t-4n}$, and so on, continuously, to build energy from echoes as they get older. In that way, almost all the energy can be regained from a reverberant environment having a single, uniform echo.

Performance of the Sigma-Delta Method

Over a finite number of echo intervals, L (during which we wait nL sample times), the energy recovered is not as much as in the infinite summations. It is only:

$$\begin{aligned} E_L &= \sum_{k=0}^{L-1} \mu^k \\ &= \left(1 - \frac{\mu}{\ln \mu}\right) + \frac{\mu^L}{L \ln \mu} \end{aligned} \quad \text{(Eq 18)}$$

If μ=0.93 and L=8, the S/N degradation would be about 1 dB, since about 88% of the energy would have been recovered.

The algorithm counts on absolute frequency and phase accuracy between transmitter and receiver. Serious phase distortion or frequency errors would render the sigma-delta method unusable. It is not well suited to SSB operation, therefore, without a pilot carrier and a synchronous (phase-locked) receiver, or other suitable demodulators.

The algorithm is also quite sensitive to phase noise in the local oscillators of radio transceivers and to dispersive propagation—the thing so similar to the problem it attempts to solve! So this algorithm turns out not to be a very good pick at all, but it is relatively simple compared to homomorphic processing.

Summary

This article showed convenient modeling methods for reverberant and dispersive environments. Systems for deconvolution were discussed that correct for multipath and dispersion; they even produce a model of the corrupting system in most cases. In some instances, the model of the corrupting system may be the thing that is sought. That is the case in ionospheric studies or in planetary science, where the impulse response of the model represents a map of the atmosphere or planetary surface, respectively. Deconvolution systems are sometimes adaptive and thus are capable of handling changing propagation conditions. Homomorphic deconvolution is generally not adaptive and relies on some knowledge of the differences between the desired signal and the nature of the medium.

Research is ongoing to use adaptive receiving arrays and homomorphic processing on weak, convolved signals. Moonbounce (EME) modes are a particular target of that research.

Notes

1. C. Hutchinson, Editor, *The 2001 ARRL Handbook*, p 18.5.
2. A. Oppenheim and R. Schafer, *Digital Signal Processing*, (Englewood Cliffs, New Jersey: Prentice-Hall, 1975).
3. B. Widrow and M. Hoff, Jr., "Adaptive switching circuits," *IRE WESCON Convention Records*, Part 4, IRE, 1960.
4. S. Reyer and D. Herschberger, "Using the LMS Algorithm for QRM and QRN Reduction," *QEX*, Sep 1992, pp 3-8.
5. M. Frerking, *Digital Signal Processing in Communications Systems*, (New York: Van Nostrand-Reinhold, 1994) p 11.
6. R. Lucky, "Techniques for Adaptive Equalization of Digital Communications Systems," *Bell Systems Technical Journal*, volume 45, pp 255-286, Feb, 1966.
7. B. Widrow and S. Stearns, *Adaptive Signal Processing*, (Prentice-Hall, 1985).
8. B. Bogert, M. Healy and J. Tukey, "The Frequency Analysis of Time Series for Echoes: Cepstrum, Pseudo-autocovariance, Cross-Cepstrum and Saphe Cracking," *Proceedings of the Symposium on Time Series Analysis*, (New York: John Wiley and Sons, 1963) pp 209-243.